Siebel

Database Upgrade Guide

April 2024

Siebel
Database Upgrade Guide

April 2024

Part Number: F84090-06

Authors: Siebel Information Development Team

Contents

Preface

This preface introduces information sources that can help you use the application and this guide.

Using Oracle Applications

To find guides for Oracle Applications, go to the Oracle Help Center at *https://docs.oracle.com/*.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the *Oracle Accessibility Program website*.

Contacting Oracle

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit *My Oracle Support* or visit *Accessible Oracle Support* if you are hearing impaired.

Comments and Suggestions

Please give us feedback about Oracle Applications Help and guides! You can send an email to: *oracle_fusion_applications_help_ww_grp@oracle.com*.

1 What's New in This Release

What’s New in This Release

This chapter tracks the changes in the documentation. It includes the following topics:

What’s New in Siebel Database Upgrade Guide, Siebel CRM 24.1 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Using REST API to Execute Schema Imports	New topic. You can manually run schema imports using the Migration Schema Service Import Business Service from inbound REST requests.

What’s New in Siebel Database Upgrade Guide, Siebel CRM 23.10 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Component Parameter to Auto-detect Updated Configuration in the Runtime Repository	New topic. Component parameter feature is introduced to allow migrated configuration to be immediately available without a user login.

What's New in Siebel Database Upgrade Guide, Siebel CRM 23.9 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Siebel CRM Upgrade Factory On Premise	New topic. Upgrade Factory On Premise enables a simpler, faster, more predictable upgrade process.
Executing a Siebel Full Migration Plan	New Topic. Executing a full migration plan using Siebel Migration.

What's New in Siebel Database Upgrade Guide, Siebel CRM 23.8 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Monthly Updates 20.8 – 22.6 to 22.7+ - Workflow Process Consideration	New Topics. Workflow Process Consideration

What's New in Siebel Database Upgrade Guide, Siebel CRM 23.7 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Executing a Siebel Migration Plan *Executing a Migration Plan*	Modified Topics. Additional information added as notes.

What's New in Siebel Database Upgrade Guide, Siebel CRM 23.6 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Siebel CRM Upgrade Factory	New topic. Siebel CRM Upgrade Factory delivers an automated development upgrade that includes the quick setup of a development environment and the efficient transition from Siebel CRM 8.0 and above to the latest Siebel CRM Release Update. The upgrade factory approach simplifies the application upgrade process, allowing customers to upload their customized, pre-IP2017 repository to run an upgrade and IRM process on Oracle Cloud Infrastructure (OCI).

What's New in Siebel Database Upgrade Guide, Siebel CRM 23.1 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Migrated Configuration Data Is Available to the Runtime Repository Without User Re-Login	New topic. It describes how the new configuration data migrated to Runtime Repository is available to users without user re-login.

What’s New in Siebel Database Upgrade Guide, Siebel CRM 22.7 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Supported Upgrade Paths for Siebel CRM	Modified topics. Updated the version numbers for supported upgrade paths.
Production Test Environment	Modified topic. There is a third task involved in upgrading a production test environment.
Production Environment	Modified topic. There is a fourth task involved in upgrading a production environment.
Logging and Diagnostics	Modified topic. Information about the TaskUpgrade utility has been added to this topic.
Running RepositoryUpgrade Utility	Modified topic. The `-9` parameter no longer defaults to "MAIN". Best practice is to specify an Integration Workspace reflecting a planned future release (you will receive an error if you specify "MAIN").
Full Runtime Repository Migration Without Using Siebel Migration	Modified topic. The last note in this topic (step 12) is new.
Managing Cross Version Migration	Modified topic. Describes how workflows and task flows are handled during cross version migration. Describes also the CleanUpTaskDR utility.

What’s New in Siebel Database Upgrade Guide, Siebel CRM 22.5 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Creating a New ODBC Data Source Before Upgrading Siebel Database	New topic. Describes how to manually create a new ODBC datasource.
Analyzing IBM DB2 Custom Tablespace Requirements for a Siebel Upgrade	Modified topic. The command line in Step 2 of this procedure has been updated to include the following switches: **`/Z UCS2_DATABASE`** and **`/A DEBUGMODE`**.
Post Installation Database Update *RepositoryUpgrade Utility*	Modified topics. The PostInstallDBSetup.zip file contains the payload for PostInstallDBSetup and the RepositoryUpgrade.zip file contains the payload for RepositoryUpgrade. Both files are located in the Siebel Server "bin" folder.

Topic	Description
Roadmap for Planning a Migration with Siebel Migration	Modified topic. The following note has been added to step 2a in this roadmap: **Note:** The process running the Siebel Object Manager must have read-write access to the network file share path for the Migration Package Location.
Configure User Access to Siebel Migration Application	New topic. To access the Siebel Migration Application, users must be assigned the appropriate responsibility (for example, "Migration Users" responsibility).
Scenario for Using a Watermark	New topic. Describes a situation when a watermark is required.
Types of Watermarks	New topic. A generated watermark file can contain several types of watermarks.
Executing a Siebel Migration Plan	Modified topic. Steps 1 and 5 have been updated.
Renaming Repositories After Full Migration	Modified topic. You must provide the full path to the siebel.cfg file (for example: **`$SIEBEL_HOME\siebsrvr\bin\enu\<siebel.cfg>`**).
Aborting a Running Migration Plan	New topic. Describes how to abort a running migration plan.
Viewing Migration Log Files	New topic. Describes how to view the log files for a migration plan.
Viewing Migration History	New topic. Describes how to view the history details for a migration plan execution.
Query Migration History	New topic. Describes how to query migration history data and filter the data shown in the History screen.
Cleanup Migration History	New topic. Describes how to cleanup migration history data by deleting history records in the History screen.
Full Runtime Repository Migration Without Using Siebel Migration	Modified topic. The last step in this procedure (re activating Task-based UI tasks in the target environment) is obsolete and has been removed.
Troubleshooting Data Migration Using Siebel Migration	New topic. This topic highlights some important issues to note when using Siebel Migration to migrate data.
Creating a New Migration Plan	Modified topic. The sample code in this topic has been updated.
About Inheriting Upgrade Behavior in a Siebel Upgrade *About the Siebel Postmerge Utilities* *CSSGridRepPatch* *CSSUINavUpgradeReposPrep* *CSSMVGUpgradePatch77*	Modified topics. These topics have been updated to remove ICL content since Siebel Incorporate Custom Layout is no longer supported.

Topic	Description
Upgrade Planning for Siebel Workflow Designer *Running the Siebel Postmerge Utilities* *Reviewing Siebel Grid-Based Applets* *Review the User Interface*	
Siebel Incorporate Custom Layout Upgrade Option (chapter) CSSWebTemplatePatch Migrating Siebel Repository Objects to the Standard User Interface Using Lag Time to Identify Postupgrade Customizations How Repository Objects are Changed How Logging is Done Specifying a Lag Time Migrating to the Standard User Interface Eliminating Obsolete Siebel UI Fields Reviewing Siebel UI Objects Affected by Incorporate Custom Layout Reviewing Required Fields in the Siebel User Interface	Obsolete topics. These topics has been removed from the guide since Siebel Incorporate Custom Layout (ICL) is no longer supported.
About Siebel Rules Expression Designer Creating Migration Rules Activating Tasks Activating Tasks with the Migration Incremental Runtime Repository Data Service	Obsolete topics. These topics have been removed from the guide.

What’s New in Siebel Database Upgrade Guide, Siebel CRM 22.3 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Marking Conflict Resolution as Complete Using Siebel Tools	Modified topic. Step 1 in this procedure has changed. $SIEBEL_HOME is the directory where Siebel Tools is installed (a typical location would be `C:\Siebel\Tools`).

Topic	Description
Renaming Repositories After Full Migration	New topic. Describes how to rename repositories after a full migration using the siebdevcli utility.
Full Runtime Repository Migration Without Using Siebel Migration	Modified topic. Steps 12 and 13 in this procedure have changed (step 14 is obsolete).
RepositoryUpgrade Utility	New topic. Describes the purpose of the RepositoryUpgrade utility and the typical order of events when planning to run the utility.
Managing Cross Version Migration	New topic. Describes how to manage some types of changes during cross version migration and the purpose of setting the FullMigCompatibilityMode system preference.
Executing a Migration Plan	Modified topic. Shows the HTTP POST request details to execute a migration plan.

What's New in Siebel Database Upgrade Guide, Siebel CRM 21.10 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Post Installation Database Update	Modified topic. The REP_VER_NUM column is automatically added to the S_WS_VERSION table during Post Installation Database Update. The purpose of the REP_VER_NUM column is to capture Siebel Repository version information when the RepositoryUpgrade utility is run – so that repository version and Workspace version are linked together.
Post Installation Database Update Exit Codes	New topic. Describes all PostInstallDBSetup utility exit codes.
Siebel Repository and Workspace Version Tracking	New topic. Describes how Siebel Repository and Workspace version tracking works.
Full Runtime Repository Migration Without Using Siebel Migration	Modified topic. Step12 in this procedure has changed.

What's New in Siebel Database Upgrade Guide, Siebel CRM 21.8 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Verifying or Editing an ODBC Definition on Windows	Modified topic. Because Siebel application uses a 32-bit ODBC data source, the registry entry for ODBC data source is as follows: **`HKEY_LOCAL_MACHINE\Software\Wow6432Node\ODBC\ODBC.INI\ODBC_Name`**.
Preparing Siebel Workflow Processes for Upgrade	Obsolete topics. These topics have been removed from the guide.

Topic	Description
About Workflow Processes in Siebel Repository	
Values Required by Post Installation Database Update	New topic. This topic replaces *Siebel Enterprise Server Installer Changes*.
Running RepositoryUpgrade Utility *Independently Apply RepositoryUpgrade Schema Changes*	Modified topics. The RepositoryUpgrade utility should always be run from the Siebel Server home directory ($SIEBEL_HOME).

What’s New in Siebel Database Upgrade Guide, Siebel CRM 21.5 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
About Migrating with Siebel Migration	The Siebel service owner account must have read-write access to Siebel File System and the Migration Package Location.
Data Migration Using Siebel Migration	Modified topic. Siebel Migration does not support migration of repository or Workspace data from RR environments to other RR environments.
Setting the Seed Migration Priority System Preference	New topic. Describes the Seed Migration Priority system preference and when to set it.
Post Installation Database Update	Modified topic. You can select whether to execute, defer, or skip running the PostInstallDBSetup utility during the update installation.
Postpone Running Post Installation Database Update	New topic. If you do not want to implement the physical database schema changes during a monthly update release but apply them later by running the PostInstallDBSetup utility, then complete the steps in this procedure.
Skip Post Installation Database Update	New topic. If you do not want the installer to run the PostInstallDBSetup utility, then complete the steps in this procedure.
Independently Apply RepositoryUpgrade Schema Changes	New topic. Describes how to apply RepositoryUpgrade schema changes external to Siebel CRM utilities.
Refreshing a Connection *Refreshing a Migration Plan*	New topics. You can refresh a connection and a migration plan for your migration by sending an HTTP POST request to the Siebel Migration Application.

What's New in Siebel Database Upgrade Guide, Siebel CRM 21.3 Update

The following table lists the changes in this revision of the documentation to support this release of the software.

Topic	Description
Upgrade Planning for Siebel AES Encryption *Modifying siebel.cfg Before Upgrading Siebel Database*	Modified topics. The siebel.cfg file must be updated correctly before running the Key Database Manager utility. For more information on the parameters to set (or update) in siebel.cfg before running the Key Database Manager utility, see *Siebel Security Guide* .
Importing with the Migration Schema Service	Modified topic. The Request Parameters (filename and password) and Request Body (filename and password) have been updated.
Importing with the Migration Application Data Service	Modified topic. The Request Parameters (filename) and Request Body (filename) have been updated.
Importing with the Migration Application Data Service With Transformation	Modified topic. The Request Parameters (filename) and Request Body (filename) have been updated.
Importing with the Migration Incremental Runtime Repository Data Service	Modified topic. The Request Parameters (filename) and Request Body (filename) have been updated.
Importing with the Migration Runtime Repository Data Service	Modified topic. The Request Parameters (filename) and Request Body (filename) have been updated.
Importing with the Migration Incremental Application Workspace Data Service	Modified topic. The Request Parameters (filename) and Request Body (filename) have been updated.
Getting a Seed Copy Import with the Migration Application Workspace Data Service	Modified topic. The Request Parameters (filename) and Request Body (filename) have been updated.
Getting the Full Seed Import with the Migration Application Workspace Data Service	Modified topic. The Request Parameters (filename) and Request Body (filename) have been updated.
Running RepositoryUpgrade Utility	New topic. Describes how to run the RepositoryUpgrade utility.
Multiple topics	In Siebel CRM 21.2 Update, the **`applicationcontainer`** directory has been replaced by two directories, as follows: • **`applicationcontainer_external`** (for Siebel Application Interface) • **`applicationcontainer_internal`** (for all other Siebel Enterprise components) In the Siebel Application Interface Installation, Web artifacts for application configurations, which were formerly located in **`applicationcontainer\webapps\siebel`**, now map to **`applicationcontainer_external\siebelwebroot`**. The **`siebelwebroot`** directory contains subdirectories such as **`files`**, **`fonts`**, **`htmltemplates`**, **`images`**, **`migration`**, **`scripts`**, and **`smc`**.

2 Overview of Siebel Database Environments

Overview of Siebel Database Environments

This chapter provides an overview of the upgrade process of the three Siebel database environments (development, test, and production). It includes the following topics:

- *Supported Upgrade Paths for Siebel CRM*
- *Terms Used in This Guide*
- *Naming Conventions Used in This Guide*
- *About File Paths and Commands in Siebel Database Upgrade Topics*
- *About Supported Siebel Upgrade Paths*
- *About Siebel Upgrade Environments*

Note: For a detailed overview of the upgrade process and related tasks, see *Overview of Performing a Siebel Database Upgrade*

Supported Upgrade Paths for Siebel CRM

The repository for the current release of Siebel CRM is SIA. Two types of optional repositories are offered while running Siebel CRM installer:

1. **Master Repository.** This is required for the following situations:
   a. You are upgrading from a pre-Siebel CRM 17.0 release.
   b. You are creating a new Siebel CRM database instance.
2. **Ancestor Repositories.** These are required if you are upgrading from a pre-Siebel CRM 17.0 release.

Review the information in the following table to determine if you need either of these repositories, and to ensure that they are available. If these repository options were not selected during installation, re-run the installer and add as needed. For more information, see *About Siebel Repositories* and *Siebel Installation Guide for the operating system you are using* .

Note: If there is a large backlog of Siebel Remote transactions in .dx files that have not been synced, then this will negatively impact the installation of monthly updates since these files will need to be copied to the backlog folder for the previous version.

Current Version	Target Upgrade Update	Upgrade Approach	Upgrade Tasks
Siebel CRM version 7.5.3 through version 7.7.2 (SEA or SIA repository)	Siebel CRM 22.7 Update	The upgrade approach is:	Perform the following two-step repository upgrade:

Current Version	Target Upgrade Update	Upgrade Approach	Upgrade Tasks
		• New installation of Siebel CRM 22.x Update for upgrade. • Two-step repository upgrade.	• Upgrade to Siebel CRM version 8.1.1 (SEA or SIA repository). • Upgrade to Siebel CRM 22.7 Update using the incremental repository merge process. For more information, see *Performing a Siebel Incremental Repository Merge*.
The current Siebel CRM version is one of the following: • 7.8.2 (SEA or SIA repository) • 8.0 (SEA or SIA repository) • 8.1.1.0 through version 8.1.1.7 (SEA repository) • 8.2 (SIA repository) • 8.2.1 (SIA repository)	Siebel CRM 22.7 Update	The upgrade approach is: • New installation of Siebel CRM 22.x Update for upgrade. • Single-step repository upgrade.	Perform the following upgrade tasks: • Run Siebel CRM 22.7 Update installer to install the 22.7 Update binaries. • Perform a Siebel Database upgrade directly to Siebel CRM 22.7. **Note:** There is no need for any intermediary steps, such as, installation of or upgrade to Siebel CRM 17.0. The New Customer Repository, generated through a three-way repository merge, contains all of the update content from Siebel CRM 22.7 Update. For more information about repository merge, see *Performing the Siebel Repository Merge*.
The current Siebel CRM version is one of the following: • 8.1.1.0 through version 8.1.1.14 (SIA repository) • 8.2.2.0 through version 8.2.2.14 (SIA repository) • 15.0 through version 15.4 • 15.5 and later patchsets • 16.0 and later patchsets of version 16.0	Siebel CRM 22.7 Update	The upgrade approach is: • Migration installation of Siebel CRM 22.x Update. • Incremental repository merge.	Perform the following upgrade tasks: • Run Siebel CRM 22.7 Update installer to install the 22.7 Update binaries. • Use incremental repository merge to bring the repository to 22.7 Update. For more information, see *Performing a Siebel Incremental Repository Merge*.
Siebel CRM version 17.0 or later	Siebel CRM 22.7 Update	Not applicable.	Neither an Incremental Repository Merge nor a Database Upgrade is required for Siebel CRM 17.0 or later (including 18.x, 19.x, 20.x, and 22.x). For more information on installing monthly updates, see the roadmap for installing Siebel CRM 22.x Update for an existing installation of Siebel CRM 17.x or later in *Siebel Installation Guide for the operating system you are using* .

Note: After installing the Siebel CRM software for the current release, you must reset any passwords stored in the Siebel Gateway that were previously encrypted using RC4 encryption. In the current release, such passwords are encrypted using Advanced Encryption Standard (AES) instead of RC4. For more information about re-encrypting passwords, see *Siebel Security Guide* .

Terms Used in This Guide

This guide uses the following terms:

- **Database Upgrade.** A set of standard procedures to migrate your custom repository and data schema from one release of Siebel CRM to a higher release level (that is, to a later release).
- **Siebel CRM Update release.** For more information about installing the current Siebel CRM Update release and about Siebel release types, see *Siebel Installation Guide* . For more information about installing Siebel Patchset releases, including new features, see *Siebel CRM Update Guide and Release Notes* on My Oracle Support, 2382435.1 (Article ID), for each applicable release.
- **SEA and SIA.** The Siebel Industry Applications (SIA) data model is physically a superset of the Siebel Enterprise Application (SEA) data model, which was formerly used. The SIA data model has more tables, more columns in the same tables, and more indexes than the SEA data model, but it does not exclude any tables, columns or indexes from the SEA version. Current versions of Siebel CRM now support only the SIA data model. Some upgrade paths might involve upgrading from an SEA data model to the SIA data model; additional steps apply.
- **Siebel Migration.** The Siebel Migration Application is a Web-based tool for end-to-end repository and data migration. Use the Siebel Migration Application to migrate the repository, runtime repository, application Workspace data, application data, application interface web artifacts and file system artifacts from a source environment to a target environment. The following topics deal with Siebel Migration:
  - *Migration Planning Using Siebel Migration*
  - *Data Preparation for Siebel Migration*
  - *Data Migration Using Siebel Migration*

  Note: The Siebel Migration Application is supported in Siebel CRM 17.0 and later releases. The Siebel Migration Application replaces the Repository Migration Utility (dev2prod, which is no longer supported).

- **Incremental Repository Merge.** A mechanism which allows you to incrementally upgrade your custom repository data (including schema and seed data) from Siebel CRM version 8.1.1.x (SIA only), version 8.2.2.x, version 15.x, or version 16.x to the latest Siebel CRM monthly update. For more information about the incremental repository merge process, see *Performing a Siebel Incremental Repository Merge*.

  Note: The Siebel CRM Update repository contains the cumulative repository, schema, and seed data for all new content (such as Release Features) developed up to the latest Siebel CRM release.

Naming Conventions Used in This Guide

This guide follows several naming conventions:

- All supported Siebel CRM releases refer to the upgrade path from which you might upgrade to a certain Siebel Business Applications release.

- For more information about your upgrade applicability and for system requirements and supported platform certifications, see the Certifications tab on My Oracle Support.

> **Note:** For Siebel CRM version 8.1.1.9 and later and version 8.2.2.2 and later, the system requirements and supported platform certifications are available from the Certifications tab on My Oracle Support. For earlier Siebel CRM versions, see Siebel System Requirements and Supported Platforms on Oracle Technology Network.

- *Current release* means the latest version of Siebel Business Applications.
- The term *Windows* refers to all Microsoft Windows operating systems listed as supported for this release in the Certifications tab on My Oracle Support.
- The term *UNIX* refers to all forms of the UNIX operating system listed as supported for this release in the Certifications tab on My Oracle Support.
- The term *IBM z/OS* refers to the IBM mainframe operating systems that are collectively referred to as z/OS and listed as supported for this release in the Certifications tab on My Oracle Support.

About File Paths and Commands in Siebel Database Upgrade Topics

Environment variables and path placeholders for both Windows and UNIX paths are used throughout *Siebel Database Upgrade Guide* . Enter UNIX commands in a Korn shell. Enter Windows commands in a Windows Command Prompt window.

Windows Paths

The following path conventions specify file system locations in *Siebel Database Upgrade Guide* topics:

- SIEBEL_ROOT is the absolute path of the Siebel Server installation directory. When you install a Siebel Server, the installation program queries for the path to the installation directory. The installation program then installs the Siebel Server in a subdirectory of this path called **`siebsrvr`**. For example, if you specified **`C:\sba81`** as the installation directory for Siebel CRM 8.1, then SIEBEL_ROOT is **`C:\sba81\siebsrvr`**.
- DBSRVR_ROOT is the absolute path to the Siebel Database Configuration Utilities on the Siebel Server. When you install the Siebel Database Server, the installation program queries for the path to the Siebel Server installation directory. The script then installs the Siebel Database Server files at the same level in a subdirectory called **`dbsrvr`**. For example, if SIEBEL_ROOT is **`C:\sba81\siebsrvr`**, then DBSRVR_ROOT is **`C:\sba81\dbsrvr`**. In this guide, many examples use the path **`C:\`** and **`C:\sba81`**.
- $SIEBEL_HOME is the directory where Siebel Tools is installed, on a developer computer running Microsoft Windows. For example, a typical location would be **`C:\Siebel\Tools`**.

UNIX Paths

The following environment variables and path conventions specify file system locations in *Siebel Database Upgrade Guide* topics:

- SIEBEL_ROOT is the absolute path of the Siebel Server installation directory and also an environment variable that defines this path. When you install a Siebel Server, the installation script queries for the path to the installation directory. The script then installs the Siebel Server in a subdirectory of this path called `siebsrvr`. For example, if you specified `/usr/siebel` as the installation directory for Siebel CRM version 8.x, then $SIEBEL_ROOT is `/usr/siebel/sba81/siebsrvr`.

  SIEBEL_ROOT and other environment variables required for doing an upgrade are located in `siebsrvr/siebenv.sh` and`siebsrvr/siebenv.csh`. The Siebel Server installation script sets environment variable definitions in these shell scripts. Do not edit or delete these files.

  > **Tip:** Before performing command-line procedures in a shell window, you must source `siebenv.csh` for a C shell or source `siebenv.sh` for a Bourne shell. Doing this refreshes the environment variables required to run commands.

- DBSRVR_ROOT is a path convention used in the *Siebel Database Upgrade Guide* . It is not an environment variable and is not defined in `siebenv.csh` or `siebenv.sh`.

  DBSRVR_ROOT is the absolute path to the Siebel Database Server files on the Siebel Server. When you install the Siebel Database Server, the installation script queries for the Siebel Server installation directory. The script then installs the Siebel Database Server files at the same level in a subdirectory called `dbsrvr`. For example, if $SIEBEL_ROOT is `/usr/siebel/sba8x/siebsrvr`, then DBSRVR_ROOT is `/usr/siebel/sba8x/dbsrvr`. In this guide, many examples use the path `/usr/siebel` and `/usr/siebel/sba8x`.
- Run UNIX scripts in a C or Korn shell. Do not run .ksh scripts in a Bourne shell.

Path Navigation

Procedural steps that ask you to navigate to a specified directory must be performed as follows:

- Windows: Open a Command Prompt window and use the cd command to make the specified directory the current directory. Do not use the Windows File Explorer to navigate to the directory. For help with the cd command, enter the word help in the Command Prompt window and press Enter.
- UNIX: In a shell window, make the specified directory the current directory.

Executing Commands

Procedural steps that ask you to execute a command must be performed as follows, unless specified otherwise:

- Windows: In a Command Prompt window, verify the current directory is correct and enter the command. Do not run the command by entering it in the Run window in the Start Menu.
- UNIX: In a shell window, verify the current directory is correct, source the siebenv script, then enter the command.

Because all versions of the UNIX operating system are case-sensitive, if you are running your Siebel Business Applications on UNIX, then treat all file names, directory names, path names, parameters, flags, and command-line commands as lowercase, unless you are instructed otherwise in the product.

If your deployment currently runs on Microsoft Windows, but you might switch to a UNIX environment or deploy UNIX servers in the future, then follow this same practice to avoid having to rename everything later.

About Supported Siebel Upgrade Paths

The supported upgrade paths for Siebel Business Applications are described in *Supported Upgrade Paths for Siebel CRM* and in the Certifications tab on My Oracle Support.

Siebel Database Upgrade Guide does not cover the following specific upgrade paths or infrastructure changes. Contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.

- Changing database platform type during an upgrade, for example changing from IBM DB2 to Oracle Database.
- Changing operating system type during an upgrade, for example changing from Windows to UNIX.
- Migrating to Unicode.
- Migrating from Oracle's Siebel Industry Solutions applications or Siebel Financial Services to Siebel Business Applications.
- Upgrading from one base language to another. To achieve similar results, upgrade your existing base language and install the language pack for the desired language.

About Siebel Upgrade Environments

There are three Siebel database environments: Development, Production Test, and Production. If you have a development environment, then you must upgrade it first. When you upgrade your development environment your previous customizations are merged with the new release. The newly merged repository and schema definitions then become inputs to the production test environment upgrade and production upgrades which you will perform after completing the development environment upgrade.

If you do not have a development environment, or have not customized your repository, then no repository merge is required. You can use the repository and schema definition files included in the new release to upgrade your production test environment and production environment.

The production environment contains only the Runtime Repository and the required Design Time Repository. The production repository is extracted during the development upgrade.

Mapping Your Environments

You might have more or fewer environments than those described in *About Siebel Upgrade Environments*. The following table gives recommendations for mapping your environments to the ones used in *Siebel Database Upgrade Guide* .

Environment Description	Recommended Upgrade
The environment has the following characteristics: • It is used primarily for development with Siebel Tools. • The Siebel database is a subset of your production database. • The environment is not used for tech support or training. Developers are usually installed as Siebel Mobile Web Clients.	Development environment upgrade.
The environment has the following characteristics: • It is intended for testing customizations before deploying them. • It is where you tune your upgrade SQL files to minimize production upgrade time. • There might be multiple upstream environments in addition to the production test environment. For example, these could include environments used by a product management group, Technical Support, and Quality Assurance.	Production test environment upgrade.
The environment is used for live business transactions by both local and remote users.	Production environment upgrade.

Siebel upgrade environments are described in detail in the following topics:

- *Development Environment*
- *Production Test Environment*
- *Production Environment*

Development Environment

This environment is where developers use Siebel Tools to customize Siebel Business Applications, and where customizations are made to the Siebel Runtime Repository. When your Siebel Business Applications instance goes live in a production environment, the Runtime Repository is where all object definitions are retrieved, then displayed to users in your Siebel CRM user interface.

A development environment includes the following Siebel CRM modules:

- Siebel Gateway
- Siebel Server

- Siebel Database Server files installed on a Siebel Server
- RDBMS server and Siebel database
- Siebel Application Interface
- Siebel Tools installed on workstations running a supported Windows environment. This includes the local database running on developers' Siebel Mobile Web Clients.
- Siebel Business Applications and test data required to verify the basic function of Siebel Runtime Repository.

Upgrading the development environment involves these tasks:

1. **Prepare application data.** These tasks prepare test data for migration.
2. **Upgrade database (upgrep).** You run the Database Configuration Wizard in upgrep mode. They perform a basic upgrade of the Siebel database schema and load repositories to prepare for the repository merge. Both upgrep and upgphys modes are run as a single step in the Database Configuration Utility.
3. **Merge repository.** You use Siebel Tools to merge your existing repository with the repository in the new release. Postmerge utilities upgrade form applets and verify that applets and views are configured correctly.
4. **Run postmerge utilities.** You use Siebel Tools to run a set of utilities that examine the merged repository. The utilities analyze your customizations and apply changes to them as needed to conform to the user interface in the new release.
5. **Upgrade database (upgphys).** You run the Database Configuration Utilities in upgphys mode. They further upgrade the Siebel database with changes resulting from the repository merge and complete the database upgrade.

   The Database Configuration Utilities also generate the customer repository definition file and logical schema definition file that are used as input to the production test environment and production upgrades.

   > **Note:** During the synchronization process, custom columns in the Siebel schema that are not in the Siebel Repository are not deleted but custom indexes in the Siebel schema that are not in the Siebel Repository are deleted.

Production Test Environment

This environment is where an installed and fully upgraded development environment undergoes the quality assurance process. This part of your upgrade process allows you to tune upgrade performance in preparation for the transition to your live, production environment.

The production test environment includes the following Siebel CRM modules:

- Siebel Gateway
- Siebel Server
- Siebel Database Server files installed on a Siebel Server
- RDBMS server and Siebel database
- Siebel Application Interface
- All the Siebel Business Applications currently installed in your production environment
- A copy of the Siebel database installed in your production environment

You perform the following processes in the production test environment:

1. Test the upgraded release to validate its function and performance before deploying it to users.
2. Tune the upgrade process to minimize the time required to perform your production upgrade.

Oracle provides an upgrade tuning application that analyzes how the upgrade scripts interacted with the production test environment database. The Upgrade Tuner enables you to tune how the scripts will execute against the Siebel database in your production environment. Tuning the scripts can significantly reduce the time required to complete the production upgrade. For this reason, the production test environment database must contain the same data volume and topography as your production database.

Upgrading the production test environment involves the following tasks:

1. **Prepare application data.** These tasks are about preparing application data for migration.
2. **Upgrade database (upgrep).** You run the Database Configuration Utilities in upgrep mode. They perform a basic upgrade of the Siebel database schema:
   - You run the utility in Prepare for Production mode before running it in upgrep mode. The Prepare for Production mode reviews the upgraded development environment database schema and creates input files. The upgrep mode uses these files to make schema changes to the Siebel database.
   - The upgrep mode imports the repository and schema definition files from the development environment upgrade. It uses these files to upgrade the Siebel database.
   - The upgphys portion runs automatically. It makes several administrative changes to table data, including updating the schema version in S_APP_VER. It does not make schema changes.
3. **Full Migration.** A one-time Full Migration is mandatory after upgrading your production test Runtime Repository (RR) environment from any pre-Siebel CRM 2017 version. This ensures that the RR environment will know which Development (Design Repository) environment provided its repository and allows it to accurately generate watermarks for future incremental migrations, as well as synchronize all administrative data (such as LOVs).

Production Environment

The production environment is your live business environment, where your internal and external users interact with applications and generate actual business data. The production environment includes all your Siebel Enterprises worldwide.

The upgrade process assumes all production environment databases are completely separate from the development environment and production test environment databases.

Oracle provides these tools to help you transition from a production test to a production environment:

- **Siebel Application Deployment Manager (ADM).** This module migrates administrative data such as lists of values (LOVs) from the production test environment to the production environment. For details, see *Siebel Application Deployment Manager Guide* .
- **Siebel Anywhere.** This application builds distribution kits for remote users. For details, see Siebel Anywhere Administration Guide.

Upgrading the production environment involves the following tasks:

1. **Additive Schema Changes.** This step is optional and is run in the production environment. You can run this mode of the Database Configuration Utilities without taking the database offline. This mode makes schema changes that do not affect the operation of the application. This reduces the amount of time the production database must be offline to perform the upgrade. Testing Additive Schema Changes is also included in the process checklist for tuning the upgrade. This allows you to verify Additive Schema Changes in the production test environment before running it in the production environment.
2. **Prepare application data.** These tasks are about preparing application data in the production database for migration.

3. **Upgrade database (upgrep).** In the production test environment, you start the Database Configuration Utilities in upgrep mode, and enter configuration information for the production environment. This includes an ODBC definition for connecting to the production Siebel database.

   This step causes the wizard to use the SQL for upgrading the production test database to upgrade the production database.
4. **Full Migration.** A one-time Full Migration is mandatory after upgrading your production Runtime Repository (RR) environment from any pre-Siebel CRM 2017 version. This ensures that the RR environment will know which Development (Design Repository) environment provided its repository and allows it to accurately generate watermarks for future incremental migrations, as well as synchronize all administrative data (such as LOVs).

The SQL generated for the production test database upgrade is preserved, and no new SQL is generated. This SQL has been tuned using the Upgrade Tuner and has been revised by the Prepare for Production mode.

You do not have to run the Database Configuration Utilities in upgphys mode. The upgphys steps are included in the production environment upgrep.

Postupgrade Tasks

Following the upgrade you perform the following tasks:

1. **Set up the environment.** These tasks set up the postupgrade environment, which includes extracting the developers' databases and running database statistics.
2. **Application administration.** These tasks set up the applications and include such tasks as setting up user access and visibility of views and screens.
3. **Application configuration.** These tasks prepare applications for testing, including data migration for specific applications.
4. **Test the system.** These tasks test the system. For development environment upgrades, you perform basic unit tests to verify the operation of the application. For production test environment upgrades, you perform a full suite of regression and stress tests to verify the system is ready for production.

3 Siebel Database Upgrade Planning

Siebel Database Upgrade Planning

This chapter provides information on planning your Siebel database upgrade. It includes the following topics:

- *About Siebel Upgrade Planning Resources*
- *Guidelines for Planning Your Siebel Database Upgrade*
- *About Upgrading Your RDBMS in the Siebel Environment*
- *About Database Sort Order in the Siebel Environment*
- *About the Siebel Database Configuration Wizard Utilities*
- *About the Siebel Upgrade Wizard and Driver Files*
- *About Siebel Additive Schema Changes Mode*
- *About the Siebel Database Upgrade Log Files*
- *About the Siebel Case Insensitivity Wizard*
- *About the Siebel Repository Merge*
- *About Siebel Repositories*
- *About Inheriting Upgrade Behavior in a Siebel Upgrade*
- *About the Siebel Postmerge Utilities*
- *About Tuning Siebel Production Upgrade Files*

Note: For a detailed overview of the upgrade process and related tasks, see *Overview of Performing a Siebel Database Upgrade*

About Siebel Upgrade Planning Resources

This topic lists important publications and resources for performing an upgrade. Review these as part of the upgrade planning process.

Product Documentation

Oracle's Siebel CRM documentation is collectively called the Siebel Bookshelf. The Siebel Bookshelf is available on Oracle Technology Network (OTN) and Oracle Software Delivery Cloud. It might also be installed locally on your intranet or on a network location.

Siebel Installation and Deployment Documentation

The following publications are meant to be used in concert with installation or deployment of the Siebel application and can be found on the Siebel Bookshelf:

- *Siebel Deployment Planning Guide*
- *Siebel Installation Guide for Microsoft Windows*
- *Siebel Installation Guide for UNIX*
- *Siebel System Administration Guide* . Provides details on how to administer, maintain, and expand your Siebel Servers.
- *Siebel Security Guide* . Provides information on the Siebel Party data model and other important security topics.
- *Siebel Application Deployment Manager Guide*
- *Siebel Performance Tuning Guide*
- *Configuring Siebel Business Applications* . Provides information about configuring Siebel Business Applications in Siebel Tools. Additionally, this guide provides information on the Siebel Party data model.
- *Siebel Data Model Reference* . The DMR describes in detail the Siebel database schema for a release. It also lists certain types of schema changes. Use the DMR during upgrade planning to evaluate how data will be stored in the new release. Consider obtaining a DMR for both the release you are upgrading from and the release you are upgrading to. Contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services for information on ordering DMRs.

Oracle Technology Network

This site contains the following:

- **Siebel Bookshelf.** A searchable collection of Oracle's Siebel CRM documentation.
- **Siebel System Requirements and Supported Platforms on Oracle Technology Network.**

Note: For releases prior to Siebel CRM version 8.1.1.9 or version 8.2.2.2, Siebel System Requirements and Supported Platforms on Oracle Technology Network is the definitive list of system requirements and supported third-party products. For all subsequent releases, the system requirements and supported platform certifications are available from the Certifications tab on My Oracle Support.

My Oracle Support

This site provides the Certifications tab and provides search access to articles, Release Notes, and various other kinds of documentation relevant to Siebel CRM:

- **Release Notes.** *Siebel CRM Update Guide and Release Notes* on My Oracle Support, 2382435.1 (Article ID), might contain late-breaking information that the *Siebel Database Upgrade Guide* does not yet include.
- **The Certifications tab on My Oracle Support.** This application is the definitive list of system requirements and supported third-party products. For more information, see the Certifications tab on My Oracle Support.

- **Articles.** Articles provide important information on specific upgrade issues. Many articles related to upgrade issues are located on My Oracle Support. References to articles are integrated throughout *Siebel Database Upgrade Guide* .
  For Siebel language support, Unicode support, and legacy code page support, see 1513102.1 (Article ID) on My Oracle Support.
- **Troubleshooting.** Troubleshooting documents contain information about how to troubleshoot common upgrade issues such as error messages encountered during the upgrade process, and other unwanted behavior in Oracle's Siebel Business Applications. Troubleshooting documents can be found for a variety of upgrade issues, including error messages found in upgrade logs. For more information browse Troubleshooting content on My Oracle Support.
- **Siebel Patchset releases.** Siebel Patchset releases and documentation are provided on My Oracle Support. For more information about installing Siebel Patchset releases, including new features, see *Siebel CRM Update Guide and Release Notes* on My Oracle Support, 2382435.1 (Article ID), for each applicable release.
  For the readme documents for Siebel Patchset releases, see *Siebel Patchset Installation Guide for Siebel CRM*, on My Oracle Support.
- **Siebel Maintenance Release Guide on My Oracle Support.** For each Siebel CRM version 8.1.1.x and version 8.2.2.x release, through Siebel CRM version 8.1.1.11 and version 8.2.2.4, Siebel Maintenance Release Guide on My Oracle Support was provided. This guide contained important information about updates to Siebel Business Applications. These guides are no longer provided for current releases.

Oracle Software Delivery Cloud

This site provides access to Siebel CRM software, including the current release of Siebel CRM. For more information about types of Siebel CRM releases, see the *Siebel Installation Guide* .

Oracle Advanced Customer Services

Contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services. These services offer detailed implementation planning and technical consulting services as well as rapid response and resolution for critical technical issues affecting Siebel deployments.

Contact your Oracle sales representative for Oracle Advanced Customer Services for important information in the following areas:

- Migrating from Siebel Industry Solutions or Siebel Financial Services to Siebel Business Applications during upgrade
- Migrating to Unicode code page support during upgrade
- Changing operating system type during upgrade

Guidelines for Planning Your Siebel Database Upgrade

This topic provides an overview of the recommended guidelines for planning upgrade resources, estimating the upgrade timeline, and managing the data migration process.

Use the following steps to help plan your upgrade.

1. **Determine your upgrade path.** Upgrade requirements and tasks differ based on the specific upgrade path involved. For more information, see *Supported Upgrade Paths for Siebel CRM*.
2. **Evaluate the complexity of the upgrade.** Determine the complexity of the upgrade effort based on the Siebel CRM modules implemented, the number of integration points, the number of interfaces, the total number of scripts, and the number of user interface scripts.
3. **Assess the current Siebel environment and evaluate the existing implementation.** Perform a detailed assessment of the current Siebel environment to determine how the implementation will be affected by the upgrade. Evaluate the current implementation in comparison with the architecture of the current release. The assessment will help you to identify areas where you can take advantage of new functionality to meet business requirements.
4. **Estimate the level of effort to upgrade.** Determine the metrics and cost associated with each aspect of the upgrade. Determine the effort required to upgrade based on the results of your complexity evaluation, current environment assessment, and new functionality review. This will help you to estimate resources, time line, and costs.
5. **Establish the upgrade team.** Assemble a cross-functional upgrade team that understands Siebel product architecture and performance guidelines. Include IT professionals, executives, and users to ensure a broad base of experience in technical, business, and Siebel-specific skills.
6. **Review interface migration tasks.** Determine the effort to migrate modified applets and views. This includes associating applets with Web template items and mapping them to Web template controls.
7. **Plan for upgrade tuning.** Tuning your production upgrade scripts can significantly reduce downtime during the final stages of your upgrade. Examples of upgrade tuning include eliminating SQL statements that do not affect any data, executing long-running SQL statements in parallel, and executing table creation, table rebuilds, and index creation in parallel.
8. **Identify data migration tasks.** After the upgrade, there might be data migration and repository configuration tasks that must be performed manually. These tasks frequently involve customizations made in prior releases.
9. **Plan for end-user training.** Analyze the impact of change on the users, and develop a plan for end-user training and adoption.

The upgrade of your application requires several key factors to be successful:

- A detailed understanding of customizations made to your current deployment
- Analysis and definition of the components within your enterprise
- Analysis of how to use new functionality provided by Oracle's Siebel software
- Strict adherence to industry best practices and guidelines identified in this guide

The upgrade planning process will produce a roadmap for the entire upgrade project that outlines infrastructure, deployment, and training requirements.

Use the results of this process to develop a project plan that identifies required skills and resources for developing and deploying the upgraded application. This will help you with advance budgeting of resources, time, and training.

Upgrade Planning Guidelines

Here are important guidelines to follow when planning an upgrade:

1. Review the Certifications tab on My Oracle Support, *Siebel Release Notes* on My Oracle Support, *Siebel Installation Guide* , and other relevant documentation. Also review the content of this guide, *Siebel Database Upgrade Guide* .
2. Gather all relevant documentation that describes the current implementation, for example, requirements documents, design documents, and architecture context diagrams.

3. Implement a change management program. For example, communicate rollout dates to users, schedule training, allow adequate time for users to adjust to the enhancements, and provide a process for end users to provide feedback to the project team.
4. User adoption is critical to a successful upgrade. Provide access to a test environment that allows users to become familiar with the new version of the application, and provide end-user training on the upgraded application.

Database Planning Guidelines

Here are important guidelines to follow when planning the upgrade of your database:

1. Analyze the impact of the upgrade on table customizations that you have made. Determine whether preupgrade data migration is required. Determine what postupgrade schema changes are required.
2. Consider database layout in your planning. Plan to tune the database and database server for the upgrade, because settings and parameters for upgrade differ from those required for Online Transaction Processing (OLTP).
3. If you are migrating multiple languages from a prior version, then plan extra time (one to two weeks) for the repository merge process. The expected merge time might increase with the number of languages in the repository. You also might need to plan for additional installation-related tasks.
4. If your locale specific language values have changed over time, then you must make changes in the upglocale.language_code file to trigger correct data migrations. For example, in a previous German language release, the German value Einzelperson mapped to the ENU value of Individual. In the new release Individuell and not Einzelperson maps to the ENU value Individual. Prior to running the upgrade utility, review the upglocale.language_code files from your present release to that of the upgrade release, to validate, and verify your language strings and to make any necessary changes. Upglocale.language_code files are located at `SIEBEL_ROOT/dbsrvr/language_code` folder in the new release's installation directory.
5. For IBM DB2, consider increasing the size of your tablespaces before going live. Make sure that your custom tablespaces are large enough for upgraded tables. See *Analyzing IBM DB2 Custom Tablespace Requirements for a Siebel Upgrade*.

Production Database Upgrade Guidelines

Both the Prepare for Production step and the upgrade tuning process modify the SQL scripts used to perform the upgrade in the production test environment. In addition, it is common to further modify these scripts to meet local requirements.

The recommended way to perform the production upgrade is to use the SQL scripts that you have generated and modified for the production test upgrade. The steps in the production environment upgrade process are as follows:

1. Run the Database Configuration Utilities in the production environment.
2. In the utilities, enter the information for the production environment instead of the production test environment. For example, you enter the ODBC connection for the production environment.

   This configures the driver file to run against the production database rather than the production test database. The utilities also configure the driver file to use the upgrade SQL files you generated for the production test upgrade.
3. Run the Upgrade Wizard. The Upgrade Wizard uses the SQL files in the production test environment to upgrade the database in the production environment.

This approach has several advantages:

- You do not have to generate upgrade SQL files in the production environment and then manually transfer customizations to them from the production test environment.
- You do not lose any changes to the SQL files that were made by Siebel Upgrade Tuner in the production test environment.
- You do not have to run the Database Configuration Utilities in Prepare for Production mode again.
- With some exceptions, you do not have to perform database-related configuration tasks required Articles or *Siebel Release Notes* on My Oracle Support.

If your network configuration prevents creating an ODBC connection to your production database from inside your production test environment, and you cannot complete your production upgrade, then create a service request (SR) on My Oracle Support. Alternatively, you can phone Global Customer Support directly to create a service request or get a status update on your current SR. Support phone numbers remain the same and are listed on My Oracle Support.

About Upgrading Your RDBMS in the Siebel Environment

Environments: Development, production test, production.

If your currently installed RDBMS version is not supported in the new release, then you must upgrade your RDBMS before performing the Siebel database upgrade. For information on supported RDBMS versions, see the Certifications tab on My Oracle Support.

If your currently installed RDBMS version is supported in the new release, and you plan to upgrade your RDBMS, then you can do so before or after upgrading your Siebel database.

The following guidelines and requirements are for planning your RDBMS upgrade.

Note: When you upgrade the RDBMS, also be sure to upgrade your client database connectivity software. See the Certifications tab on My Oracle Support.

Oracle Database

The following guidelines and requirements are for planning your upgrade on Oracle Database.

- If you upgrade the Siebel database and then upgrade the Oracle RDBMS, then you must validate the Siebel database schema after the upgrade. See *Verifying an Upgraded Oracle Database After a Siebel Upgrade*.
- Development, production test, and production upgrades on Oracle Database 11*g*, can be run in CBO mode.
- If you plan to upgrade your Oracle Database, then see 781927.1 (Article ID) on My Oracle Support. This document was previously published as Siebel Technical Note 582. This document provides information on setting up Oracle Database for use with Oracle Cost-based optimizer (CBO) and on doing other performance tuning for Oracle Database. Using Oracle's Cost Based Optimization with a Siebel database entails certain requirements for database parameter configuration and for the Optimizer's statistics generation on tables, columns and indexes.

IBM DB2

The following guidelines and requirements are for planning your upgrade on IBM DB2.

- After upgrading your IBM DB2 RDBMS, you must upgrade the database instance.
- The IBM DB2 database must have 4-KB, 16-KB, and 32-KB tablespaces defined on it. Otherwise, your upgrade will not complete successfully.

About Database Sort Order in the Siebel Environment

Environments: Development, production test, production.

Sort order (also called collation sequence) is specified during the initial installation of a database and defines the way in which the database sorts character data when executing queries. Sort order support depends on both the code page of the database and whether it is used in a development or a production environment:

- **Development environments.** For development databases, you must use a binary sort order due to the functional limitations of databases that use a nonbinary sort order.
- **Production environments.** For production databases, it is strongly recommended that you use binary sort order to prevent possible performance degradation.

The settings for binary sort order are unique for each database platform. For Siebel language support, Unicode support, and legacy code page support, see 1513102.1 (Article ID) on My Oracle Support.

Sort Order Considerations for Siebel Databases

If your deployment requires that you use a nonbinary sort order (for example, if your local language does not use binary sort order), then you must consider several functional limitations that particularly affect development environment upgrades. If these limitations are unacceptable, then consider re-creating your database to use binary sort order. However, be aware of these considerations:

- You cannot use Siebel Tools to create or update a Siebel Runtime Repository on a database that uses a nonbinary sort order.
- You cannot perform a repository merge on a database that uses a nonbinary sort order.
- Databases that use nonbinary sort order might perform slower than databases that use binary sort order.

For Siebel language support, Unicode support, and legacy code page support, see 1513102.1 (Article ID) on My Oracle Support.

About the Siebel Database Configuration Wizard Utilities

Environments: Development, production test, production.

The Database Configuration Utilities are part of the Siebel Database Configuration Wizard. The Database Configuration Utilities interactively gather the information required to perform the following operations:

- **Install the Siebel Database.** These wizards set up the Siebel database in the RDBMS as part of a first-time installation of Siebel Business Applications. They can also add a language to the Siebel Database Server installation.
- **Upgrade the Siebel Database.** These wizards upgrade the Siebel database to a new release in a development environment, production test environment, or production environment.
- **Import or export a Siebel repository.** These wizards move entire repositories between database environments with the same schema definition. A wizard that adds installed languages to a Siebel Tools repository is also provided.
- **Apply Additive Schema Changes.** As part of upgrading to a new release, this wizard generates a SQL script. The script contains schema changes that you can make to the production database without taking it offline. This reduces the production database downtime required for the upgrade.
- **Run database utilities.** This group of wizards perform the following functions:
  - Synchronize a database schema with the Siebel Repository schema definition.
  - Convert existing time-keeping management to UTC.
  - Convert existing Lists of Values (LOVs) to Multilingual Lists of Values (MLOVs).

Note: You must migrate Siebel repositories using the Siebel Migration Application.

About the Upgrade the Siebel Database Function

When you run the Upgrade the Database function of the Database Configuration Wizard, the wizard prompts you for the upgrade environment (development or production) and the upgrade phase (upgrep or upgphys). The wizard then prompts you for the information required to perform the upgrade.

After collecting and confirming the information, the wizard creates an upgrade configuration file and calls a driver that uses the file to set up the SQL scripts required to upgrade your database.

After you run the Database Configuration Wizard, you run the Upgrade Wizard. The Upgrade Wizard opens a state file containing the steps for the upgrade. These steps execute the SQL scripts created after you ran the Database Configuration Wizard.

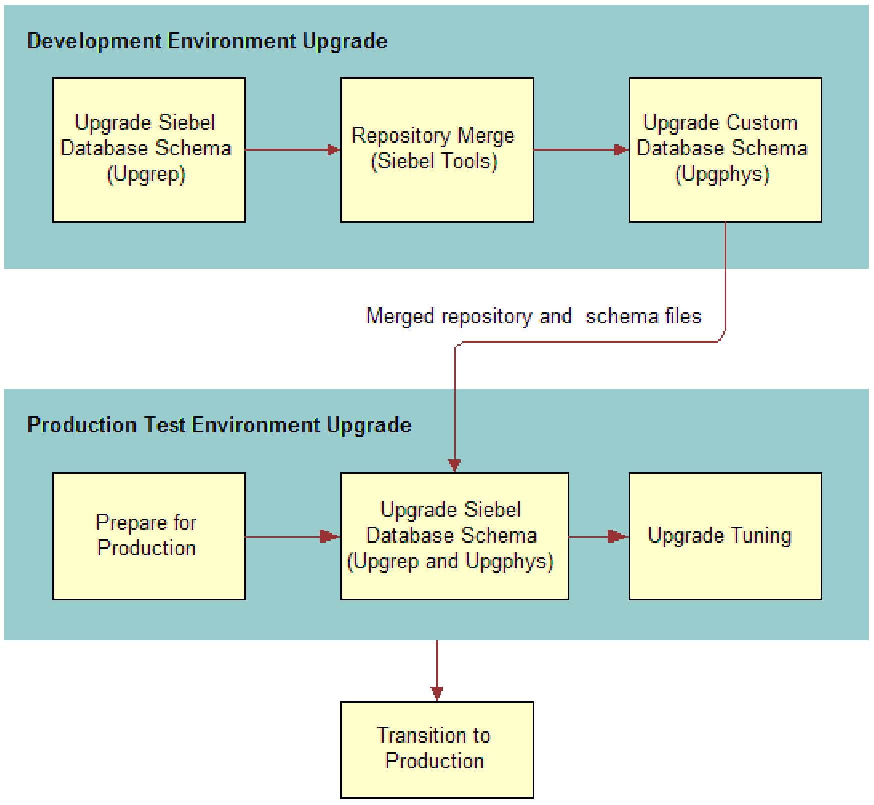
To upgrade a development environment, production test environment, and production environment, you run the Database Configuration Wizard (and Upgrade Wizard) several times, as listed in the following table.

Upgrade Step	Select This Environment Type	Select This Upgrade Option
Development environment upgrep	Development	upgrep
Development environment upgphys	Development	upgphys
Production test environment prepare for production	Production	Prepare for Production

Upgrade Step	Select This Environment Type	Select This Upgrade Option
Production test environment upgrep and upgphys	Production	upgrep and upgphys
Apply Additive Schema Changes	Production	Apply Additive Schema Changes
Production environment upgrep and upgphys	Production	upgrep and upgphys

The following image shows how the Database Configuration Wizard (and Upgrade Wizard) work together with the Siebel Tools repository merge to upgrade your environments. The steps shown in the following image are as follows:

1. Development Environment Upgrade:
   a. Upgrade Siebel database schema (Upgrep)
   b. Repository merge (Siebel Tools)
   c. Upgrade custom database schema (Upgphys)
2. Merged repository and schema files from development environment are sent to production test environment.
3. Production Test Environment Upgrade:
   a. Prepare for production
   b. Upgrade Siebel database schema (Upgrep and Upgphys)
   c. Upgrade tuning.
4. Transition to Production.


Development Environment Upgrade
Upgrade Siebel Database Schema (Upgrep)
Repository Merge (Siebel Tools)
Upgrade Custom Database Schema (Upgphys)
Merged repository and schema files
Production Test Environment Upgrade
Prepare for Production
Upgrade Siebel Database Schema (Upgrep and Upgphys)
Upgrade Tuning
Transition to Production

Development Environment Upgrades

For a development environment upgrade, you run the utility twice, once in each of the following modes:

1. **upgrep.** This mode makes the following changes:

    - Removes interface tables and database triggers.
    - Populates columns that must change from NULL to NOT NULL.
    - Creates new tables. Merges existing tables.
    - Prepares for index creation. Verifies that there are no unique key violations.
    - Creates indexes.

      > **Note:** Indexes that have been previously deactivated will be recreated during the upgrep phase. Check to determine whether these indexes are required following the development upgrade. If they are not, then they might be deactivated again. To deactivate indexes, contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.

    - Imports seed data.
    - Imports the Prior Siebel Repository, New Siebel Repository, and New Customer Repository.
    - Makes modifications to repository objects to prepare for the repository merge.
    - Updates primary children foreign key references.
    - Performs miscellaneous file actions.

2. **upgphys.** This mode makes the following changes:

    - Synchronizes the Siebel database schema to the logical schema definition in the merged repository.
    - Deduplicates intersection tables.
    - Exports repository object definitions to a file, custrep.dat.
    - Exports the logical schema definition to a file, schema.ddl.

      These two files are used as input to the production test environment and production environment upgrades.
    - Renames the New Customer Repository to Siebel Repository.
    - Updates the schema version in S_APP_VER.

Production Test Environment Upgrades

For a production test environment upgrade, you run the utility three times:

1. **Prepare for production.**

   > **Note:** This topic is only applicable for upgrades from Siebel CRM release 7.5.3. For upgrades from later releases, you do not have to perform this step.

   This mode does the following in the production test database:

   - Examines the development environment database and generates SQL that deduplicates intersection tables and sets up support for database aggregation.

     In the production test environment, you must define an ODBC connection to the development environment database before performing this upgrade step.
   - Compares the repository schema and the physical database schema. Generates a file that lists indexes present in the physical schema that are not present in the repository schema. You can decide whether to remove these indexes.
2. **upgrep.** This mode makes the following changes:

   - Removes interface tables and database triggers
   - Populates columns that must change from NULL to NOT NULL
   - Uses the custrep.dat and schema.ddl files from the development environment upgrade to create new tables and merge existing tables
   - Prepares for index creation. Verifies that there are no unique key violations.
   - Creates indexes
   - Imports seed data
   - Updates primary children foreign key references
   - Performs miscellaneous file actions
   - The upgphys portion runs automatically. It makes several administrative changes to table data, including updating the schema version in S_APP_VER.
3. **Upgrade Tuning.** These steps are optional. You can use the Logparse utility to tune the SQL generated for the upgrade. Logparse identifies SQL commands that are not used, or that can be run in parallel. This reduces the time required to perform the production database upgrade.

You can also test the Additive Schema Changes script to verify that it does not adversely affect the operation of the application. Additive Schema Changes allows you to run part of the schema upgrade without taking the database offline. This reduces production database downtime.

Production Environment Upgrades

After you have completed testing applications and have tuned the upgrade SQL commands, you perform the production upgrade. The production upgrade uses the SQL commands generated in the production test environment.

In the production test environment, you perform the following steps to upgrade your production environment Siebel database:

1. **ODBC Connection.** Define an ODBC connection to the production environment Siebel database
2. **Apply Additive Schema Changes.** This step is optional and is run in the production environment. You can run this mode of the Database Configuration Utilities without taking the database offline. This mode makes schema changes that do not affect the operation of the application. This reduces the amount of time the production database must be offline to perform the upgrade.
3. **Prepare for Production.** This step is not required. You ran it as part of the production test environment upgrade. The required changes have already been made to the upgrade SQL commands.
4. **upgrep.** Run the Database Configuration Utilities in the production test environment. Enter information for the production environment (not the production test environment), including the new ODBC definition.

   The Database Configuration Utilities update the upgrade configuration file with production environment information. A lock file that was set when you ran the utility previously, prevents new SQL from being generated. This preserves the SQL you have revised and tuned.

   When you run the Upgrade Wizard, it reads the production environment information from the configuration file and uses the production test environment SQL commands to upgrade the production environment Siebel database.

   The upgrep step makes the same changes as when it ran in the production test environment. This includes automatically running the production upgphys portion of the upgrade.

How the Upgrade Configuration File and SQL Files Are Created

When you run the Database Configuration Utilities, the following actions are performed:

- Collect configuration information.
- Create a primary upgrade configuration file (UCF). This file maps the information you entered in the Database Configuration Utilities to environment variables.
- Forward the information to an SQL generator that creates or populates SQL files with the required commands. The SQL generator extracts these commands from an intermediate XML file containing all the SQL commands required for an upgrade.
- Prompt you to start the Upgrade Wizard.

In some cases, you must modify the generated SQL files as required by Articles, or *Siebel Release Notes* on My Oracle Support before you run the Upgrade Wizard. To do this, answer No when prompted to run the Upgrade Wizard. Then, edit the SQL files and manually launch the Upgrade Wizard.

How to Locate Master Configuration Files

Primary upgrade configuration files are stored in the following location:

- Windows: `SIEBEL_ROOT\bin`
- UNIX: `$SIEBEL_ROOT/bin`

Primary upgrade configuration files use the following naming convention:

```
master_UPGRADEOPTION_ENVIRONMENT_VERSION.ucf
```

where:

- UPGRADEOPTION indicates the upgrade process you are performing
  - `upgrep` indicates Siebel database schema upgrade
  - `upgphys` indicates Custom database schema upgrade
  - `prepare_for_production_upgrade` indicates Prepare for Production upgrade
- ENVIRONMENT indicates the environment that you are upgrading
  - `dev` upgrades Development environment
  - `prod` upgrades Production environment
- VERSION indicates the version from which you are upgrading

For example, if you are upgrading from Siebel CRM version 8.1.1, then the primary upgrade configuration file generated from the development environment upgrep is as follows:

```
master_upgrep_dev_811.ucf
```

The primary upgrade configuration file generated from the Prepare for Production mode is as follows:

```
master_prepare_for_production_upgrade.ucf
```

Related Topics

About Siebel Additive Schema Changes Mode

About the Siebel Upgrade Wizard and Driver Files

About the Siebel Upgrade Wizard and Driver Files

Environments: Development, production test, production.

The Upgrade Wizard performs the upgrade. It uses the information collected by the Database Configuration Utilities to execute commands and SQL scripts. These commands and SQL scripts are contained in driver files. Driver files are in ASCII text format and are organized into steps. The Upgrade Wizard reads the steps in the driver files and performs the commands contained in each step.

In a driver file, steps are separated by a blank line, and each step begins with a File Execute Entry number. The key part of each step is the command or script to be executed. When an SQL script is specified, you can review the script and see what changes it will make to the Siebel database before running the Upgrade Wizard.

Driver files are provided for each of the major upgrade operations. Examples of driver files are as follows:

- `driver_upgrep_dev_811.ucf`
- `driver_upgphys_dev_811.ucf`
- `driver_upgrep_prod_811.ucf`

Here is an excerpt from a driver file that controls a development environment upgrep from Siebel CRM *version number* on Oracle Database. The excerpt contains three steps:

```
[Sql File Entry 0]
Type = DbSql
File Name = rename_existing_repositories.sql
```

```
Use Table Owner = 1
Use System Admin = 0
IgnoreAllDDLErrors = 0
IgnoreAllDMLErrors = 0
Argument 0 = $SiebelVersion
Title = Verify Repository Name
Title Message Num = 0
Estimated Disk Space = 0
Backup Db = 0
Parallelizable Item = 0
Prompt User For Status = 0
[File Execute Entry 1]

Type = FileExecute
File Name = $SiebelRoot\bin\odbcsql
Check Return Code = 1
Return Code Compliance = 0
16 Bit App = 0
Command Line = /s "$ODBCDataSource" /u $TableOwner /p $TablePassword /separator / /
a /c rem /l $SiebelLogDir/dropif-db.log $DbsrvrRoot/$DatabasePlatform/dropif-db.sql
/v
Number of 10 Second Wait Loops = 2000
Return Code = 0
Title = Drop interface tables
Title Message Num = 0
Estimated Disk Space = 0
Backup Db = 0
Parallelizable Item = 0
Prompt User For Status = 0
[File Execute Entry 2]

Type = FileExecute
File Name = $SiebelRoot\bin\odbcsql
Check Return Code = 1
Return Code Compliance = 0
16 Bit App = 0
Command Line = /s "$ODBCDataSource" /u $TableOwner /p $TablePassword /separator / /
a /c rem /l $SiebelLogDir/trigdrop-db.log $DbsrvrRoot/$DatabasePlatform/trigdrop-
db.sql /v
Number of 10 Second Wait Loops = 2000
Return Code = 0
Title = Drop triggers
Title Message Num = 0
Estimated Disk Space = 0
Backup Db = 0
Parallelizable Item = 0
Prompt User For Status = 0

[File Execute Entry 2]
```

How to Locate Upgrade Driver Files and SQL Scripts

Driver files are stored in the following location:

- Windows: `DBSRVR_ROOT\PLATFORM\UPGRADE\Version_Number`
- UNIX: `DBSRVR_ROOT/bin/PLATFORM/UPGRADE/Version_Number`

where:

- PLATFORM indicates the database type, for example ORACLE
- VERSION indicates the version from which you are upgrading

For example, if you are upgrading from Siebel CRM version 8.1.1, then the driver file for the development environment upgrep will be:

```
driver_upgrep_dev_811.ucf
```

Most of the SQL scripts generated or populated for the upgrade are stored in the same directory as the driver file. The remaining SQL scripts are stored in the PLATFORM directory.

Related Topics

About the Siebel Database Configuration Wizard Utilities

About the Siebel Repository Merge

About Siebel Additive Schema Changes Mode

Environments: Production test, production.

Apply Additive Schema Changes mode is part of the Database Configuration Utilities. The purpose of Additive Schema Changes is to reduce the downtime required for the production database upgrade.

Additive Schema Changes generates an SQL script. You run the script on the live production database before you take it offline to perform the production upgrep. The script performs certain upgrade steps that are normally part of the production upgrep. This reduces the number of steps the upgrep must perform while the database is offline, and thus reduces database downtime.

Running the SQL script is optional. If you do not run it, then the regular production upgrep performs the steps in the SQL script.

Types of Changes

Additive Schema Changes make the following types of schema changes to support the new release. These changes do not adversely affect data integrity or database normalization:

- Creating new tables.
- Adding columns to an existing table. The column must either be specified as null, or if the column is not null, it must have a specified default value.
- Increasing column sizes for numeric or varchar columns. The column must not be the basis for a picklist. Also, the resultant cumulative row size must not be larger than the data page size.
- Revising a not-null column to null.
- Revising data type from char to varchar.

Implementation of Additive Schema Changes

You generate the Additive Schema Changes script by starting the Database Configuration Utilities and choosing Apply Additive Schema Changes. A wizard prompts you for environment information similar to other upgrade modes.

After you review and confirm your entries, the wizard populates the file `platform\driver_additive_gen.ucf`. In this path, platform is the database type (DB2, MSSQL, or Oracle).

The Upgrade Wizard then performs the following steps:

- Reads the `master_additive_gen.ucf` file to obtain environment information and the name of the driver file.
- Reads the `driver_additive_gen.ucf` file and executes the ddlimp2 command it contains.
- The ddlimp2 command performs the following steps:
  - Reads the tables and indexes from the `schema.ddl` file.
  - Connects to the specified database and reads the existing schema and indexes.
  - Identifies the upgrade schema changes that can be made with the database online.
  - Writes these upgrade changes to the `schema.additive.sql` script.
  - Writes all actions to `schema.additive.log`.

The Upgrade Wizard does not execute `schema.additive.sql` against the database.

How to Use Additive Schema Changes

Use the following workflow to implement Additive Schema Changes:

1. In the production test environment, complete all development and upgrade testing. The `schema.ddl` file must be in its final form.
2. As part of running the Tuning Upgrade Performance process, restore the production test environment database to its original, unupgraded state.
3. Run the Database Configuration Utilities in Apply Additive Schema Changes mode, and generate the `schema.additive.sql` script.
4. **Windows.** You will not be prompted whether you want to run the Upgrade Wizard. Instead, the Upgrade Wizard starts automatically, and creates the `schema.additive.sql script`. The Upgrade Wizard does not run the script against the database.
5. **UNIX.** After the Database Configuration Utilities exit, run the Upgrade Wizard to generate the `schema.additive.sql` script. The Upgrade Wizard does not run the script against the database.
6. Review the `schema.additive.log` file. Verify that the script was created successfully.
7. Review the `schema.additive.sql` script. Verify that it will not make changes that conflict with customizations. Edit the script as required.
8. Use an SQL editor to apply the `schema.additive.sql` script to the database.
9. Verify that applications function normally. If necessary, revise `schema.additive.sql` and reapply it to the database.
10. Complete the remainder of the Tuning Upgrade Performance process. See *Process of Tuning Siebel Upgrade Performance*.
11. When you are ready to upgrade the production database, run the `schema.additive.sql` script against the production database before doing the production upgrep.
12. There is no need to then run database statistics. The schema changes are not useful for optimizing applications function before the upgrade.

The steps in this workflow have been incorporated into the process you use to perform both upgrade tuning and the production environment upgrade.

Related Topic

About the Siebel Database Configuration Wizard Utilities

About the Siebel Database Upgrade Log Files

Environments: Development, production test, production.

Platforms: Windows and UNIX only. Does not apply to IBM z/OS.

The Upgrade Wizard writes log files that provide detailed information on the upgrade processes, for example development-environment upgrep and upgphys. These log files provide detailed information on the upgrade steps, and they also list all errors. The Logparse utility analyzes and summarizes these log files.

Upgrade Wizard Log Files

The Upgrade Wizard writes the log files for a process to the following directory:

- Windows: `SIEBEL_ROOT\log\PROCESS`
- UNIX: `$SIEBEL_ROOT/log/PROCESS`

In these paths, PROCESS specifies `UPGRADEOPTION_ENVIRONMENT_VERSION` by default.

For example, the log files for a Siebel development environment upgrep process appear in the following locations:

- Windows: `SIEBEL_ROOT\log\upgrep_dev_versionnumber`
- UNIX: `$SIEBEL_ROOT/log/upgrep_dev_versionnumber`

You can select a different directory for the log files, by specifying a value for the Log Output Directory option in the Database Configuration Wizard. This option is for specifying a different subdirectory, under the `log` directory, in which to create the log files.

Each subdirectory in the `log` directory contains the following files:

- **output.** This subdirectory contains the Upgrade Wizard log files.
    - `upgwiz.log` (`srvrupgwiz.log` on UNIX). The Upgrade Wizard log for the process. This log contains detailed information on how the Upgrade Wizard ran. If you run the process more than once, the log name increments, for example `upgwiz2.log` or `srvrupgwiz2.log`.
    - Windows only. Log files with the name `upgwizn(nnn)` can be ignored. For example, `upgwiz1(001).log`, `upgwiz1(002).log`, and so on.
    - Windows only. Log files with the name `sw_cfg_xxx` can be ignored. (These are located in the `log` directory.)
    - UNIX only. Log files with the name `srvrupgwizn_nnn` can be ignored. For example, `srvrupgwiz1_01.log`, `srvrupgwiz1_02.log`, and so on.
    - Log files with the name `siebel.log` can be ignored.

  Tip: Each upgrade process, for example, a development upgrep, has a driver file located in `DBSRVR_ROOT\platform\upgrade\version` . An example of a driver file: `driver_upgrep_dev_811ucf`. The Upgrade Wizard performs the steps in the driver file. If a step calls a utility, such as ddlimp or dataimp, then the `/l` option in the command syntax specifies the name of the log file. Unless you modified the output location for the log file, it is located at `siebsrvr\log\process` .

- **state.** This subdirectory contains the state.log file.
    - **state.log.** The state.log file lists each step in the upgrade process and whether the step completed successfully.

      The output and state log file directories are automatically archived on subsequent runs of a process that completes successfully. (The names of subsequent log directories are appended with `_1`, `_2`, and so on.) To preserve disk space, periodically delete or save log directories to another location.
    - **summary.** This subdirectory is generated by the Logparse utility. The Logparse utility also generates a report that summarizes the upgrade, summary.html. The summary.html file contains links to the html pages in the `summary` subdirectory.
    - **summary.html.** This HTML file is created by the Logparse utility. It summarizes the log files in the `output`directory. Use this file to verify that all upgrade steps completed successfully and to determine whether there were any errors.
    - **summary.txt.** This text file is created by the Logparse utility. This file contains the same information as summary.html.

      Note: If a browser is not installed on the computer you are using, then review summary.txt instead of summary.html.
    - **summary.xml.** This file is produced by the Logparse utility during production upgrades and used by the Siebel Upgrade Tuner to parallelize table creation and index creation and to inactivate SQLs that affect no rows. (Development upgrades do not produce a summary.xml file.)
    - **upgtuner_ftp_get.txt. (UNIX Only).** This file is produced by the Logparse utility during production upgrades. You can use this file to transfer the upgrade scripts from your UNIX computer to a Windows temporary directory to perform upgrade tuning. (Development upgrades do not produce `upgtuner_ftp_get.txt`.)
    - **upgtuner_ftp_put.txt. (UNIX Only).** This file is produced by the Logparse utility during production upgrades. After you tune your production upgrade scripts, you can use this file to transfer the upgrade scripts from the temporary directory on your Windows computer back to your UNIX environment. (Development upgrades do not produce `upgtuner_ftp_put.txt`.)

The Logparse Utility

For each upgrade process, the Logparse utility analyzes the log files in `siebsrvr\log\process` . It then summarizes the log files and provides several reports. The Logparse reports provide the following information.

Parameters Report

The Parameters report lists the total time required for the upgrade process. It also lists environment variables and input files used by the upgrade process. Use this report to verify that the upgrade was set up correctly.

Step/Error Summary

The Step/Error Summary lists each step in the upgrade process and whether it completed successfully. It also provides a link to a detailed step summary for each step.

The detailed step summary lists database errors returned by the Upgrade Wizard, SQL scripts, and errors returned by the upgrade utilities called by the driver files, such as ddlimp, and dataimp.

For Oracle Database and IBM DB2, the detailed step summary lists only unacceptable errors that must be corrected. The summary does not list benign errors.

For Microsoft SQL Server, the detailed step summary lists all errors, including benign errors. An errors file (`errors.xls and errors.rtf`) is provided that lists benign errors for each database type. When you find an error in the summary.html file, you must compare it with the error file to determine whether it is benign.

Performance Report

The performance report provides a list of the longest running SQL scripts and a list of SQL scripts returning no rows. The lists include a link to each script. Upgrade Tuner uses this information as input to help you tune upgrade performance.

The report also provides two lists of DDL commands executed during the upgrade process. The DDL Summary lists the tables that were created, altered, or deleted by the ddlimp utility. The Native DDL Summary lists tables that were created, altered, or deleted by SQL scripts and includes a link to each script.

About the Siebel Case Insensitivity Wizard

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

Databases: All databases.

Query features provide indexes that directly support case and accent insensitive (CIAI) queries on eligible text columns. The Siebel Case Insensitivity Wizard configures specified columns for CIAI queries by defining CIAI columns and CIAI indexes in the repository. The wizard also sets the Default Insensitivity property for these columns to DB Case & Accent.

The purpose of the enhanced CIAI features is to improve query effectiveness and performance. Running the Case Insensitivity Wizard is optional.

Note: Before performing an upgrade or incremental repository merge, you must drop or remove any CIAI indexes and CIAI columns. Run the Case Insensitivity Wizard again after completing the upgrade or incremental repository merge.

Related Topic

Siebel Case Insensitivity Wizard

Related Book

Configuring Siebel Business Applications

About the Siebel Repository Merge

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

Repositories are located in tables in the Siebel database. These tables store the definitions of the objects used to build Siebel Business Applications. These tables also store Siebel schema definitions. When you display objects in the Siebel Tools Object List Explorer, you are displaying records from these tables.

The repository tables contain three types of records:

- Object definitions used to create Siebel Business Applications, such as business components, applets, controls, and the relationships between them.
- Definitions of the tables in the Siebel database (metadata). These records define the logical schema of the Siebel database. Later in the upgrade process, you will synchronize the physical schema of the Siebel database with the logical schema as defined by these records.
- Runtime Repository tables. The Runtime Repository tables usually begin with S_RR prefix and store the compiled definitions of the repository objects.

Repository records include a repository ID that identifies the repository to which the record belongs. The repository ID forms part of the user index for records and allows several repositories to be stored in the same set of tables.

The Siebel Repository is the deployed, active Siebel Tools repository. You use this repository to customize applications.

The development upgrep adds three additional repositories to the Siebel database, as listed in the following table. These repositories are required for the repository merge.

Repository Name	Added During Upgrade?	Siebel Tools Name	Description
Prior Customer Repository	No	Prior Customized	This is your current Siebel Repository, before upgrading. It contains any changes you have made. You renamed it Prior Customer Repository before doing the initial database upgrade.
Prior Siebel Repository	Yes	Prior Standard	The standard Siebel Repository for your installed release (the release you are upgrading from).

Repository Name	Added During Upgrade?	Siebel Tools Name	Description
			Is called the common ancestor repository in the upgrade SQL scripts.
New Siebel Repository	Yes	New Standard	The Siebel Repository for the release to which you are upgrading. This repository defines the new release.
New Customer Repository	Yes	New Customized	A second copy of the New Siebel Repository. Your customizations from the Prior Customer Repository are merged into this repository. After the upgrade, this becomes the Siebel Repository.

What Happens During a Repository Merge?

The repository merge process identifies differences between the repository in your old release (Prior Customer Repository) and the repository in the new release (New Siebel Repository). The merge process then merges these differences into the New Customer Repository. The merge process searches for the following types of objects in the Prior Customer Repository: customer-created, customer-deleted, and customer-modified.

Customer-Created Objects

Customer-created objects are high-level (or first-level) objects, such as screens, applets, and business components, that you have created in the Prior Customer Repository. If an object is present in the Prior Customer Repository but not in the Prior Siebel Repository, then it is customer-created. The merge process copies customer-created objects intact to the New Customer Repository.

Customer-Deleted Objects

Customer-deleted objects are high-level objects you have deleted in the Prior Customer Repository. If an object is absent in the Prior Customer Repository but present in the Prior Siebel Repository, then it is customer-deleted.

If you delete a high-level object in the Prior Customer Repository and it is present in the New Customer Repository, then the merge process does not delete the object from the New Customer Repository. After the merge, you must review these objects and remove them as desired.

If you delete a high-level object from the Prior Customer Repository, and it is obsolete (inactive) in the New Customer Repository, then the object will be present and inactive in the New Customer Repository.

Customer-Modified Objects

A customer-modified object has the following characteristics:

- It is a high-level object, such as screen, applet, or business component.
- The object exists in the Prior Siebel Repository and the Prior Customer Repository. (The object is not customer-created or customer-deleted.)
- The properties of the object in the Prior Customer Repository and Prior Siebel Repository are not the same. (You have modified the object.)

If the object properties are also different between the Prior Siebel Repository and New Siebel Repository, then the object has changed in the new release, and a conflict exists. The merge process logs the conflict and resolves it. After the merge, you must review how these conflicts were resolved and change the resolutions as desired.

For customer-modified objects where no conflict exists, the merge process copies the modifications to the object to the New Customer Repository.

If you modify a high-level object by deleting any of its child objects, then the merge process does not delete these child objects in the New Customer Repository. After the merge, you must review child objects and remove them as desired.

Some of the child objects of Applet are an exception. If you delete the following child objects of Applet from the Prior Customer Repository, then the merge process deletes these objects from the applet in the New Customer Repository:

- Control
- List Column
- Page Tab
- Chart
- Applet Web Template Item
- View Web Template Item

For example, if you delete a button from Applet-A in the Prior Customer Repository. The merge process deletes this button from Applet-A in the New Customer Repository.

The following table shows how a regular merge handles both customer-modified and customer-created objects. The columns list the status of a repository object:

- **Standard.** The object appears in the Prior Standard Repository, and in the New Siebel Repository, and is not customer-modified.
- **Deleted.** You have deleted the object from the Prior Customer Repository (customer-deleted).
- **Customized.** You have modified the object in the Prior Customer Repository (customer-modified).
- **Revised.** The object has changed in the new release (New Siebel Repository).
- **New.** You have created the object in the Prior Customer Repository (customer-created), or the object is new in the new release (New Siebel Repository).
- **Inactive.** The object is present in the New Siebel Repository and New Customer Repository but is inactive and not used in the new release. The object is obsolete.

The first three columns list the status of the object in the three repositories that the merge process compares during the merge. The last column, Merged New Customer Repository, lists the status of all high-level repository object types after the repository merge is complete and the postmerge utilities have run.

Prior Standard Repository (PSR)	Prior Customer Repository (PCR)	New Siebel Repository (NSR)	Merged New Customer Repository
Standard	Standard	Standard	Standard
Standard	Standard	Revised	Revised
Standard	Standard	Standard, or Inactive	Standard, or Inactive

Prior Standard Repository (PSR)	Prior Customer Repository (PCR)	New Siebel Repository (NSR)	Merged New Customer Repository
Standard	Customized	Customized	Customized
Standard	Customized	Customized, or Inactive	Customized, or Inactive
Standard	Customized	Revised (conflict)	Revised (conflict)
Not applicable	New	New	New
Not applicable	Not applicable	New	New
Standard	Deleted	Standard	Standard
Standard	Deleted	Not applicable	Not applicable
Standard	Deleted	Not applicable	Not applicable

Upgrade Ancestor Property

You can link repository objects together so that one object inherits the upgrade behavior of another. You do this by specifying an upgrade ancestor for an object. You can specify upgrade ancestors for the following object types:

- Applets
- Business Components
- Integration Objects
- Reports

For example, you create Applet-B by copying Oracle standard Applet-A. In Applet-B you specify Applet-A as the upgrade ancestor. In the New Siebel Repository, Applet-A has a new button. Because Applet-B is a descendant of Applet-A, the merge process adds the new button to both Applet-A and Applet-B.

If you had not specified Applet-A as an upgrade ancestor of Applet-B, then Applet-B would not have received the new button.

Postmerge Utilities

The postmerge utilities run after the repository merge completes. The utilities make changes to objects in the New Customer Repository. The purpose of the postmerge utilities is to reduce the configuration changes required as part of reviewing applications and user interfaces after the merge.

Related Topics

About Inheriting Upgrade Behavior in a Siebel Upgrade

About the Siebel Postmerge Utilities

Reviewing Siebel Repository Object Property Conflicts

About Siebel Repositories

Prior to the Siebel CRM version 8.1.1, two Siebel repositories supported Siebel applications: Siebel Business Applications (SEA) repository and Siebel Industry Applications (SIA) repository. Deploying applications from outside either group was not possible through a direct, single-step upgrade process.

Since Siebel CRM version 8.1.1, a single repository is supported (SIA) and an upgrade process is available that enables a direct, single-step upgrade from SEA to SIA repositories. After running database configuration utilities, you can perform direct upgrades from an SEA deployment to a higher version SIA repository through a single-step process, for certain supported upgrade paths. For example, you can directly upgrade to the current release from Siebel CRM version 7.8 SEA or SIA, or from version 8.0 SEA or SIA, using a single-step upgrade method.

Data Model Changes for Repository Upgrade

Due to data model changes around Address objects between the SEA and SIA repositories, the migration from the SEA repository to the SIA repository will result in some SEA objects being made obsolete. You must reconfigure all Address-related business components that are in use. In general, the same development upgrade process needs to be applied for the SEA to SIA migration on the same version. Migrating from SEA to SIA on the same version requires less effort then upgrading from one version to a higher version. For more information, see *Migrating Address Data After a Direct SEA to SIA Upgrade*.

About Inheriting Upgrade Behavior in a Siebel Upgrade

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

You can link objects together so that one object inherits the upgrade behavior of another. You do this by specifying an upgrade ancestor for an object.

Use standard objects as upgrade ancestors. A *standard object* is an uncustomized repository object provided by Oracle.

A common customization strategy is to create new objects by making a copy of a standard object and then modifying the copy, called the *descendant*.

For descendants that are for the following object types, you can specify an upgrade ancestor:

- Applets
- Business Components
- Integration Objects

- Reports

You specify the upgrade ancestor in the descendant's Upgrade Ancestor field in Siebel Tools. During the repository merge, the descendant is upgraded in the same way as the upgrade ancestor.

For example, you create Applet-B by copying standard Applet-A. In Applet-B you specify Applet-A as the upgrade ancestor. In the New Siebel Repository, Applet-A has a new button.

After a regular merge, both Applet-A and Applet-B will have the new button.

Limitations on the Upgrade Ancestor Property

The Upgrade Ancestor property is considered only during repository merges as part of application upgrades under these conditions:

- If an upgrade ancestor is inactive in the New Siebel Repository, then it is obsolete, and its upgrade behavior is not propagated to descendants.

  If an ancestor object is obsolete in the New Siebel Repository, then its descendants are not also obsolete.

  If an upgrade ancestor is not present in the New Siebel Repository, then error messages display during the repository merge and are written to the merge log file. These errors are acceptable and do not mean the merge has failed.
- The Upgrade Ancestor property is not considered during repository imports. However, imported objects can specify an upgrade ancestor. When the next application upgrade is done, the Upgrade Ancestor property is taken into account.
- The setting of the Upgrade Ancestor property is not considered when applying application patches. If the upgrade ancestor is modified by the patch, then descendants are not modified.
- Use caution when specifying upgrade ancestors. For regular merges, setting the Upgrade Ancestor property on applets propagates merge problems from standard-object applets to descendant applets.
- Specifying an upgrade ancestor for objects slows the repository merge.

Upgrade Ancestor Picklist

When you click in the Upgrade Ancestor field, a picklist displays. The following criteria are used to populate the picklist:

Applets

- Table is the same as the current applet buscomp
- Class is the same as the current applet class
- Upgrade Ancestor is null
- Applet is a standard object

Reports

- Buscomp is the same as the current report buscomp
- Class is the same as the current report class
- Upgrade Ancestor is null

- Report is a standard object

Business Components

- Bus Comp is the same as the current business component
- Class is the same as the current business component
- Upgrade Ancestor is null
- Business component is a standard object

Integration Objects

- Base Object Type is the same as the current Base Object Type
- Business Object is the same as the current business object
- Upgrade Ancestor is null
- Integration object is a standard object

Propagating Changes to Objects After the Merge

If you do not select an upgrade ancestor for an object, then changes to the upgrade ancestor are not propagated to the descendant during the repository merge.

You can manually propagate changes to descendants after the merge by using the Siebel Tools object comparison and synchronization features. These features enable you to compare any two objects and propagate differences to one or both of the objects. For more information, see *Configuring Siebel Business Applications* and *About the Siebel Repository Merge*.

About the Siebel Postmerge Utilities

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

You run the postmerge utilities after the repository merge completes.

The postmerge utilities make revisions to objects in the New Customer Repository. The utilities are useful for identifying problems with user interface customizations.

The postmerge utilities are implemented as a framework. Each utility in the framework is a business service and is listed in reputility.xml:

Windows: `$SIEBEL_HOME\reppatch\reputility.xml`

The postmerge utilities are as follows:

- **CSSGridRepPatch.** Converts customized applet controls to grid-based layout.
- **CSSUINavUpgradeReposPrep.** Converts customized screens to the Siebel CRM version 7.7 user interface navigation scheme.

- **CSSMVGUpgradePatch77.** Enables the shuttle feature in customized MVGs.
- **CSSUIUpgradeReports**. Lists unresolved problems in converting customized screens and views to the Siebel CRM version 7.7 user interface navigation scheme.
- **CSSWFRepPatch.** For workflows, this utility changes step references from row-id references to name references for child objects of steps.

CSSGridRepPatch

The layout of form applets is grid-based, rather than flow-based. Applet Form Web templates have several new properties: Grid Property, Row Span, and Col Span. In addition, the item identifier syntax changes to xxyyy, where xx is the grid row and yyy is the grid column where the control is located.

Regular merge. The utility identifies both customer-created and customer-modified applets and adjusts the properties of controls in them as follows:

- If you have modified the location of an existing control, then the utility restores the control to its original location. If you have modified other properties of the control, then these changes are preserved. You must move the control and label to the desired location.
- If you have added a new control, it does not display after the merge. To display the control, the utility assigns an item identifier that places the control in the lower portion of the applet. The utility creates a Grid Property, Row Span, and Col Span property for the item. The utility also creates a Label control. You must move the new control and label to the desired location.
- The utility does not change property settings for controls that are unmodified.

CSSUINavUpgradeReposPrep

At Siebel CRM version 7.7, a declarative model for associating views with screens was introduced. The relationship between views and screens for all levels of navigation must be explicitly declared. Several new object properties for screens and views are introduced to support this.

The new navigation scheme is applied to all screens and views for regular merges. The utility runs after the repository merge in all upgrades. However, it is intended primarily for upgrades from releases prior to Siebel CRM version 7.7.

After a regular merge, screens and views display as follows:

- Unmodified screens display normally.
- The views in customer-created screens do not display.
- Views added to existing screens do not display.

The utility scans all screen view definitions looking for orphaned views. It groups orphaned views under existing Categories. If no Category exists, then the utility creates one. This causes the orphaned views to display.

For a regular merge, you must review customer-created and customer-modified screens and views to verify that views are correctly associated with screens.

The utility generates output to the reputility.log section called User Interface Navigation Upgrade.

CSSMVGUpgradePatch77

MVG applets with an M:M relationship to the underlying business component are configured as shuttle applets by default. The utility scans these MVG applets in the New Customer Repository and reconfigures them to display as shuttle applets:

- **Regular merge.** After the merge, customer-created and customer-modified MVGs are not shuttle-enabled. The utility reconfigures these MVGs so they display as shuttle applets.

The utility generates output to the reputility.log section called Multi Value Group Shuttle Applet Upgrade.

CSSUIUpgradeReports

This utility makes no changes to the New Customer Repository. Instead, it scans user interface objects in the repository and lists problems that could not be resolved by the CSSUINavUpgradeReposPrep utility.

The CSSUIUpgradeReports utility writes the report to reputility.log. You must manually correct problems listed in the report. The report is located in the POST MERGE USER INTERFACE REPORTING UTILITY section of the log.

The report has the following sections:

- Issue 1: Rich Text Control (RTC) that needs to have User Properties Reconfigured
- Issue 2: New Aggregate Category Records that must be renamed
- Issue 3: Views that need an applet in View Web Template Item Id 1
- Issue 4: Chart Views Needing Migration to Aggregate Type
- Issue 5: Explorer Views Needing Migration to Aggregate Type
- Issue 6: Categories where parent applets are missing drilldowns to a Detail View

CSSWFRepPatch

This utility changes workflow step references from row-id references to name references for child objects of steps. This completes the process of migrating workflows to the repository at Siebel CRM version 7.7. This utility primarily affects upgrades from releases prior to Siebel CRM version 7.7.

This utility does not write to the reputility.log.

How the Postmerge Utilities Work the Upgrade Behavior Property

The postmerge utilities ignore the Upgrade Behavior property. They make changes to user interface objects based on the object’s characteristics rather than Upgrade Behavior setting.

Related Topics

About the Siebel Repository Merge

About Tuning Siebel Production Upgrade Files

Environments: Production test environment only. Does not apply to production environment.

Platforms: Windows and UNIX only.

The Upgrade Tuner allows you to tune the upgrade files generated by the Siebel Database Configuration Utilities. Tuning the production upgrep files can significantly reduce database downtime when performing a production environment upgrade.

Upgrade Tuner displays the following information and provides the following options:

- **Table creation times.** You can place tables with long creation times in parallel threads.
- **Index creation times.** You can assign index creation to run in parallel threads.
- **Zero-Row SQL commands.** You can review and inactivate SQL statements that do not affect any table rows.

The scripts used to upgrade your Siebel database are generic. They update your Siebel database to support all Siebel Business Applications functionality. You can use this option to eliminate SQL statements that are not needed for your applications.

Operating System and RDBMS Support

Upgrade Tuner supports parallel threads for table and index creation for the combinations of operating system and RDBMS shown in the following table.

Operating System	Oracle Database	IBM DB2	Microsoft SQL Server
Windows	Yes	No	No
UNIX	Yes	No	No

Upgrade Tuner is not supported for IBM z/OS. Upgrade Tuner supports zero-row SQL command deactivation for the combinations of operating system and RDBMS shown in the following table.

Operating System	Oracle Database	IBM DB2	Microsoft SQL Server
Windows	Yes	Yes	Yes
UNIX	Yes	Yes	No
z/OS	Not applicable	No	Not applicable

For information on which versions and releases of the operating systems and RDBMS products that Oracle supports for Siebel products, see the Certifications tab on My Oracle Support.

Upgrade Tuner is part of the Siebel Server and runs only under Microsoft Windows. Upgrade Tuner does not run on UNIX. To tune UNIX production upgrade files, you must copy them to a Windows computer, tune them, and move them back to the UNIX computer. Scripts are provided to move the files.

If you are a UNIX customer and do not have a Siebel Server for Windows, then contact your account manager or Oracle Global Customer Support to obtain one.

When to Use Upgrade Tuner

Use Upgrade Tuner in the production test environment to tune the upgrade files that perform the production upgrep. There is no need to tune the upgrade files that perform the production upgphys. You also do not need to tune the upgrade files that perform the development environment upgrep or upgphys.

Upgrade Tuner Modes

When you start Upgrade Tuner it displays four tabs:

- Process Information
- Parallelize Table Creation
- Parallelize Index Creation
- Deactivate 0-Row SQLs

Process Information Tab

This page displays the information sources that Upgrade Tuner is using. These sources include the Logparse summary.xml file, the primary .ucf file, and the driver .ucf file.

You cannot edit the information on this page. Upgrade Tuner obtains the information by reading the summary.xml file.

Parallelize Table Creation and Parallelize Index Creation Tabs

The Parallelize Table Creation page and the Parallelize Index Creation page both have the same layout. These pages allow you to do the following:

- **Parallelize Table Creation tab.** This page displays the time required to create tables and allows you to assign table creation to parallel threads. Adding a table to a parallel thread does not add index creation for that table to the thread. Table and index creation are handled as separate steps during the upgrade.
- **Parallelize Index Creation tab.** This page displays the time required to create table indexes and allows you to assign index creation to parallel threads.

Creating parallel threads improves upgrade performance by reducing the total time to create tables and indexes. You can create up to nine parallel threads. Tables or indexes not assigned to a parallel thread remain in the serial thread.

When you run the Upgrade Tuner, the Number of Threads parameter is automatically set, based on a log analysis, to the value effectively required to run the upgrade. In most cases this value is set at 4. This value, which can be changed in the master_upgrep* file, drives the maximum number of threads for each single Siebel upgrep task, such as ddlimp executions concurrently running for this process. When setting this value you must consider the system resources of the computer from which the upgrep phase was initiated. These changes must be tested on a copy of the Siebel production database environment, prior to rolling these out for the actual Siebel production upgrade downtime window.

Deactivate 0-Row SQLs

This page allows you to activate or deactivate the SQL statements that do not affect any table rows and therefore any data. This capability improves the upgrade performance by eliminating SQL statements that might not apply to your data. This page lists only the SQL files that are executed natively by the RDBMS. It does not list SQL files that are executed using odbcsql.

Files Required to Run Upgrade Tuner

Upgrade Tuner requires the files listed in the following table. The location of the files is the same on both Windows and UNIX hosts.

File	Location on a Windows Host
`summary.xml`	`SIEBEL_ROOT\log\upgrep_prod_VERSION` For example, `SIEBEL_ROOT\log\upgrep_prod_version\summary.xml`
`master_upgrep_prod_VERSION.ucf`	`SIEBEL_ROOT\bin` For example, `SIEBEL_ROOT\bin\master_upgrep_prod_version.ucf`
`schema*.ddl`	DBSRVR_ROOT\DBPLATFORM For example, `DBSRVR_ROOT\Oracle\schema.ddl, schema_i1.ddl, schema_t1.ddl`
`driver_upgrep_prod_VERSION.ucf`	`\DBSRVR_ROOT\DBPLATFORM\upgrade\VERSION` For example, `DBSRVR_ROOT\Oracle\upgrade\version\driver_upgrep_prod_version.ucf`
`*.sql`	`DBSRVR_ROOT\DBPLATFORM\upgrade\VERSION` For example, `DBSRVR_ROOT\Oracle\upgrade\version\pret.sql, preschm.sql`

How Upgrade Tuner Modifies Files

When you save your changes, Upgrade Tuner modifies the upgrade files. These are the files the Upgrade Wizard uses to upgrade the database. These files are as follows:

- The driver configuration file.
- The schema.dll file.
- SQL files.

Driver Configuration File

When you add or remove parallel threads and save your changes, Upgrade Tuner modifies the driver configuration file, for example `driver_upgrep_version.ucf`. The driver configuration file is a text file that contains a series of steps. The steps specify the commands that control the production upgrep. When you run the Upgrade Wizard to upgrade your database, it executes the steps in the driver configuration file.

The following actions are examples of steps that can appear in the driver configuration file:

- Making schema changes using the ddlimp utility and schema.ddl
- Making schema and data changes by executing SQL scripts

Upgrade Tuner manages the driver configuration file as follows:

- When you save your changes after the first session, Upgrade Tuner makes a copy of the file and appends `.orig` to the file name. It then modifies the file. For example, Upgrade Tuner copies `driver_upgrep_version.ucf` to `driver_upgrep_version.ucf.orig`. It then makes changes to `driver_upgrep_version.ucf`.
- When you save your changes after the second session, Upgrade Tuner makes a copy of the file and appends `.old` to the file name. It then modifies the driver file. For example, Upgrade Tuner copies `driver_upgrep_version.ucf` to `driver_upgrep_version.ucf.old`. It then makes changes to `driver_upgrep_version.ucf`.
- When you save your changes after the third session and all following sessions, Upgrade Tuner saves the driver file to `.old` again and then updates the driver file.

This file management strategy preserves the previous set of revisions to the file. It also preserves the original version of the file.

Schema.ddl File

When you run the Upgrade Wizard after the production upgrep it reads the driver file. The driver file contains steps that call the ddlimp utility. This utility uses schema.ddl as input to upgrade your database schema.

In the Parallelize Table Creation and Parallelize Index Creation pages, Upgrade Tuner displays the creation times for the tables and indexes in the schema.ddl file.

When you create parallel threads, Upgrade Tuner creates thread-files that have the same format as schema.ddl but contain only the table or index creation steps in the thread. Upgrade Tuner then adds steps to the driver file. These steps call the ddlimp utility, and specify the thread-files as input.

Upgrade Tuner manages schema.ddl and thread-files as follows:

- When you create a new thread and click Save and Exit, Upgrade Tuner creates a schema.ddl thread-file for the new thread.

  For example, you do not have any parallel threads, and then create two new threads for table creation. When you exit, Upgrade Tuner creates a `schema_t1.ddl` and a `schema_t2.ddl` file. Upgrade Tuner also inserts steps in the driver file to execute the thread-files.

The t1 thread-file contains the table creation information for the tables in Parallel Thread 1. Parallel thread 2 information is contained in the t2 thread-file, and so on.

- When you create new threads in the Parallelize Index Creation page, the thread-files are named i1, i2, and so on. For example, the information for Parallel Thread 1 for index creation is contained in `schema_i1.ddl`.

- If you run Upgrade Tuner and change the tables or indexes assigned to a thread, then Upgrade Tuner updates the thread-file for that thread. Upgrade Tuner does not create .orig or .old files for thread-files. Also, Upgrade Tuner does not change the step that executes the thread-file in the driver file.
- Upgrade Tuner does not revise the content of the main schema.ddl file (the serial thread) when you create thread-files. The thread-files duplicate the content in schema.ddl.

In the driver file, the order of execution of steps for schema.ddl and the thread-files is as follows:

- Table thread files beginning with file t1 (ddlimp in table creation mode)
- schema.ddl (ddlimp in table creation mode)
- Index thread files beginning with i1 (ddlimp in index creation mode)
- schema.ddl in index creation mode (ddlimp in index creation mode)

For both table and index creation, the parallel threads are executed first followed by the serial thread (schema.ddl).

Tip: To locate thread-file steps in the `driver_upgrep_prod_version.ucf` file, query for `"schema_"`.

SQL Files

When you make changes in the Deactivate 0-Row SQLs page, Upgrade Tuner makes changes to the SQL file containing the SQL command. Because the SQL file is already a step in the driver file, Upgrade Tuner does not modify the driver file.

Upgrade Tuner manages the SQL files as follows:

- When you first change an SQL file, Upgrade Tuner saves a copy of the SQL file and appends .orig to its file name. Upgrade Tuner then updates the SQL file.
- The next time you change the SQL file in Upgrade Tuner, it saves a copy of the SQL file and appends .old to the file name. Upgrade Tuner then updates the SQL file.
- Thereafter, when you modify the SQL file, Upgrade Tuner saves the SQL file to .old again and then updates the SQL file.

This file management strategy preserves the previous set of revisions to the file. It also preserves the original version of the file.

4 Application Planning for a Siebel Upgrade

Application Planning for a Siebel Upgrade

This chapter provides information on Siebel Business Applications and issues to consider when upgrading your Siebel database. This chapter includes the following topics:

- *Upgrade Planning for Multilingual Siebel Deployments*
- *Upgrade Planning for Language String Migration*
- *Upgrade Planning for Siebel Unicode Support*
- *Upgrade Planning for Siebel AES Encryption*
- *Upgrade Planning for Siebel Web Template Files and Style Sheets*
- *Upgrade Planning for Siebel Access Control*
- *Upgrade Planning for Migrating Siebel Address Data*
- *Upgrade Planning for Siebel Workflow Designer*
- *Upgrade Planning for Mobile Devices in the Siebel Environment*
- *Upgrade Planning for Resonate Central Dispatch in the Siebel Environment*
- *Upgrade Planning for Siebel String Translation*
- *Upgrade Planning for Siebel Personalization*
- *Upgrade Planning for Siebel Pricer and Order Management*

Upgrade Planning for Multilingual Siebel Deployments

Environments: Development, production test, production.

Platforms: Windows, UNIX, IBM z/OS.

The upgrade process upgrades both the primary (base) language and all deployed languages.

> **CAUTION:** Each multilingual deployment must contain a primary language as well as those that are designated as secondary, or non-primary. For instance, a deployment might have ENU as the primary language and FRA, and JPN as its non-primary languages. Replacing the primary language, ENU in this example, with one of the non-primary languages during upgrade is not supported, as it will result in severe data corruption, making recovery extremely difficult.

The development and production upgrep process does the following for the primary language and all deployed languages:

- (Development upgrep). Imports user interface strings into the Prior Standard Repository, New Customer Repository, and New Siebel Repository.
- Imports seed data into the database.

- In the S_LST_OF_VAL table, an SQL script activates all newly imported records and configures them to support multilingual lists of values (MLOVs). LOVs remain enabled for the primary language.

No language-related manual tasks are required to upgrade the deployed languages, except where noted.

Validation

Prior to starting a development or production upgrep, you must have installed the new release's language packs for all deployed languages.

The Database Configuration Utilities validate deployed languages by comparing the status of language packs installed on the Siebel Server with the language IDs of records in the S_LST_OF_VAL table. If there are language IDs in S_LST_OF_VAL but no corresponding deployed language on the Siebel Server, then the validation fails and you cannot continue the upgrade.

Prior to starting a development or production upgrep, you must review the records in S_LST_OF_VAL, using Siebel Tools. Remove or deactivate records for languages that you do not have deployed.

For more information, see *Running the Siebel Database Configuration Wizard on Windows* or *Running the Siebel Database Configuration Wizard on UNIX*, depending on your operating system.

CAUTION: Regardless of whether you have American-English deployed, there will be records in S_LST_OF_VAL where the Language Name value equals English-American. Do not deactivate or delete these records. The validation process ignores them.

Restrictions

The following restrictions apply to multilingual upgrades:

- Installed languages must be deployed. If you have installed a language but not deployed it, then the language will not be upgraded.

A deployed language is one where you have installed the language pack, imported the user interface strings into the Siebel Repository, imported the seed data into the database, activated the seed data records added to S_LST_OF_VAL, and configured the records for MLOVs.

- You cannot select which languages to upgrade. The Upgrade Wizard upgrades all deployed languages.
- You cannot add or remove a language during an upgrade. If you want to add a language that is not currently deployed, then do so after the upgrade. For more information on adding new languages, see *Siebel Global Deployment Guide* and *Siebel Installation Guide* .

Upgrade Planning for Language String Migration

Upgrades from: All Supported Siebel releases.

Environments: Development, production test, production.

Databases: All databases.

Locale-specific language values can change from one release of Siebel Business Applications to another. Some ENU values shipped with your upgrade might no longer map to language strings from your previous release. For instance, in a German environment, the English word Individual mapped in previous releases to the German word Einzelperson, but in the subsequent release now maps to the German word Individuell. To ensure correct data migration, it is advised that you review your previous upglocale.lang_code file against the current file of the same name to determine whether any language strings require changes to ensure that they properly map when the upgrade utilities are run.

Upgrade Planning for Siebel Unicode Support

Environments: Development, production test, production.

For Western European languages and Japanese, Siebel Business Applications support both non-Unicode and Unicode code pages. For all other supported languages, Siebel Business Applications support only Unicode code pages.

Unicode is recommended if your installation uses Siebel Email Response, correspondence, or similar functionality, particularly if content is generated on a separate system.

This is because Siebel Business Applications use Unicode internally. If the RDBMS is not using Unicode, then Siebel Business Applications convert content from Unicode to the database code page. Using a Unicode code page for the database prevents string conversion problems on text content.

Before converting to Unicode, your encryption method must be AES.

CAUTION: Migrating to Unicode is more complex than simply importing your existing data into a Unicode database. Failure to execute the migration correctly can result in serious data corruption or unrecoverable data loss. For this reason, Oracle's Advanced Customer Services participation is mandatory. To perform a Unicode migration, contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.

For Siebel language support, Unicode support, and legacy code page support, see 1513102.1 (Article ID) on My Oracle Support.

For information about Unicode and global deployment for Siebel Business Applications, see *Siebel Global Deployment Guide* .

For information on upgrading to AES encryption, see *Siebel Security Guide* .

Planning Considerations for the Unicode Migration

Migrations to Unicode require the assistance of Oracle's Advanced Customer Services.

Contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services to migrate your upgraded database from a non-Unicode code page to Unicode. To perform the migration, you can use either database vendor native utilities or Siebel utilities.

If you are planning to migrate your upgraded application to Unicode, then consider the following points:

- **Database size increase. Migration to Unicode increases the size of your database.** For this reason, you must allocate additional space for your database before migrating to Unicode. For more information, create a service request (SR) on My Oracle Support. Also, you can contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.

- **IBM DB2 data truncation.** Migration to Unicode might cause truncation of certain data in IBM DB2 databases. In the past, long columns with a type of varchar could have a maximum length of 16,383 characters. However, in Unicode, the maximum length of long columns with a type of varchar is 16,350. During the migration to Unicode, long columns of type varchar that exceed 16,350 are truncated. To prevent this, you can perform tasks to identify which data might be truncated and take appropriate measures before migration. For more information, create a service request (SR) on My Oracle Support. Also, you can contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services. See also *Identifying IBM DB2 Long Columns for Truncation in a Siebel Upgrade*.
- **IBM DB2 custom tablespace information.** The upgrade does not preserve custom tablespace information for IBM DB2 databases. This presents a problem during your migration to a Unicode code page, because you must know which tables need to be re-created. You must modify the upgrade scripts to handle custom tablespaces.

  For instructions about how to modify upgrade scripts to handle custom tablespaces, contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.
- **Third-party product integration.** Migration to Unicode might affect integration with third-party systems. For more information, create a service request (SR) on My Oracle Support. Also, you can contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.
- **Environment code pages.** You cannot use a non-Unicode code page for your development environment, and then later migrate to Unicode for your production environment.

Supported Types of Unicode

The current release supports two types of Unicode:

- **UTF-8.** UTF-8 uses the same encoding for Western European languages. It occupies one byte for Western European languages and up to three bytes for some Asian languages, such as Japanese.
- **UCS-2.** UCS-2 is supported for IBM DB2 and Microsoft SQL Server databases. UCS-2 does not map one-to-one with Western European languages. It occupies two bytes for all languages.

Upgrade Planning for Siebel AES Encryption

Upgrades from: All Supported Siebel releases.

Environments: Development, production test, production.

Databases: All databases.

Platforms: Windows and UNIX only.

Siebel CRM 7.7 introduced support for the AES encryption method, the government standard for secure applications. Siebel CRM no longer supports RC2 encryption but continues to support AES encryption.

Data that is unencrypted or that uses the standard encryptor (supported in some earlier releases) or RC2 cannot be read by the application in the current release so you must upgrade your encryption method to AES using the Encryption Upgrade Utility. Running the Encryption Upgrade Utility encrypts data that is unencrypted and increases the encryption level of data that is already encrypted. For more information about upgrading data to a higher encryption level, see *Siebel Security Guide* .

Note: All encryption that is upgraded is upgraded to a minimum of 256 bits in Siebel CRM.

During the upgrade process, the encryption upgrade is run during the database upgrep and upgphys process. During the database upgrep upgrade, only the logical definition is updated. During the database upgphys process, only the physical data is updated. During the upghys encryption upgrade, the .cfg file could cause errors. The .cfg file must be updated correctly before running the Key Database Manager utility - for more information, see *Modifying siebel.cfg Before Upgrading Siebel Database*.

Note: After installing the Siebel CRM software for the current release, you must reset any passwords stored in the Siebel Gateway that were previously encrypted using RC4 encryption. In the current release, such passwords are encrypted using Advanced Encryption Standard (AES) instead of RC4. For more information about reencrypting passwords, see *Siebel Security Guide* .

Upgrade Planning for Siebel Web Template Files and Style Sheets

Environments: Development environment only.

Siebel Web templates and Siebel Web template files (SWT files) help define the layout and formatting of the user interface, such as views, applets, and controls. The Siebel Web Engine in the Siebel Server uses the Siebel Web templates to build Web pages, which it then forwards to the Siebel Application Interface.

A Siebel Web template file contains regular HTML or XML tags interspersed with Siebel tags. Siebel tags are prefixed by `swe` and contain placeholders for user interface objects such as controls and data. HTML formatting tags are defined in cascading style sheets (such as main.css).

When you install Siebel Application Interface or Siebel Tools, you receive new style sheet files. The upgrade process does not use your existing files. If you have manually customized your Siebel Web template files or style sheet files, then you must evaluate whether to reimplement these customizations in the new Siebel Web template files and style sheet file.

Observe the following planning guidelines for reimplementing customizations:

- Resolve any user interface problems related to object definitions in the Siebel Tools repository first. While doing so, review the areas of the new user interface where you plan to implement Siebel Web template file customizations.
- Evaluate existing customizations to Siebel Web template files, and decide which ones to reimplement. Changes to the user interface in the new release might make some customizations obsolete.
- Document why each customization was reimplemented. This reference will help you evaluate customization issues later.
- Use formal change control for managing versions of the Siebel Web template files and style sheet files. This allows you to maintain orderly distribution of the files to developers.

- Applets typically have a separate Siebel Web template file for each applet mode. Customize all the mode Siebel Web template files for an applet at the same time. This allows you to verify applet functionality in a single test pass in the user interface.
- Individual Siebel Web template files are typically used by multiple screens, views, or applets. Set up formal test plans that verify customizations are correct across all user interface objects that use each Siebel Web template file. This reduces the amount of time required to verify customizations and prevents unintended changes to the user interface.
- Reimplement style sheet customizations in two passes. On the first pass, implement only those changes required to address user interface usability issues. On the second pass, implement the remaining style sheet customizations after Siebel Web template customizations are complete. This shortens the time required to resolve functional problems in the user interface.

All Siebel Web template files must be copied into one directory and you must provide the same folder path during the upgrade process. The style sheet files must be copied to the Siebel Application Interface installations in the environment. The files must also be included in upgrade kits sent to remote sites.

Siebel Web template files have an `.swt` file extension and are located in the `webtempl` directory in both the Siebel Server and Siebel Tools installations.

The style sheet files are located in the installation directory of Siebel Application Interface and Siebel Tools (Windows path syntax). For example:

Siebel Application Interface: `SIEBEL_AI_ROOT\applicationcontainer_external\siebelwebroot\files`

Siebel Tools: `$SIEBEL_HOME\public\files\main.css`

The steps for reimplementing customizations to these files and copying them to new environments are included in the upgrade process topics in *Overview of Performing a Siebel Database Upgrade*

For information on how Siebel Web templates and style sheet files work, see *Configuring Siebel Business Applications* and Configuring Siebel Open UI.

In Siebel CRM 15.x and later, a script is executed that migrates data from Siebel Web template files into table-based content in the Siebel database. For Incremental Repository Merge, Siebel Web template migration occurs during the upgrep operation. For full upgrades, this migration occurs during the upgphys operation.

For more information, see *Converting Siebel Web Templates with the SWT to OD Conversion Utility* .

Upgrade Planning for Siebel Access Control

Environments: Development, production test, production.

Platforms: Windows, UNIX, IBM z/OS.

Access control refers to all mechanisms that control visibility of screens, views, and data within Siebel Business Applications. Access control includes, but is not limited to, positions, responsibilities, organizations, and access groups.

To implement access control within your Siebel Business Applications, your Siebel administrator creates relationships between people and resources (a more general term for data that includes views and functionality). These relationships or policies are authorizations. Both people and resources can be grouped and placed in hierarchies to simplify administration.

External users, such as customers and channel partners, can be assigned varying access levels that control visibility of data and application functionality. When planning access policies, consider the following:

- The complexity of access control policies (one data item or group of data items can be accessed by one or many users or groups, but not by all).
- The amount of content that is distributed by the Siebel Business Applications, including Master data (data that is static and referential, such as Products) and Customer data (data that is created and managed by users of applications, such as Opportunities).
- The number of users and entities that access the data. Also consider the complexity of relationships between users (partners, competitors, browsers, customers).

For more information on access control, see *Siebel Security Guide* .

Upgrade Planning for Migrating Siebel Address Data

In previous releases, address data was stored as follows:

- The relationship between person and address was 1:M and was stored in the table S_ADDR_PER.
- The relationship between account and address was 1:M and was stored in S_ADDR_ORG.
- Both tables included a column ADDR_NAME, which is a computed value based on other attributes in the address table.
- The user key for S_ADDR_PER included PER_ID and ADDR_NAME.

How Address Data Is Preserved

Because PER_ID is no longer part of the user key for S_ADDR_PER, the ADDR_NAME must be unique for all records.

It is possible that records within or across S_ADDR_ORG and S_ADDR_PER could have the same ADDR_NAME. If this occurs, then the ADDR_NAME for one of the records is preserved, and the upgrade process appends the ROW_ID to ADDR_NAME for the others. This prevents records from being deleted and preserves all records from both tables.

How to Manage Address Migration

You must perform the following tasks to migrate address data:

- Before upgrading the database, you must run a script to identify records that have the same ROW_ID between S_ADDR_PER and S_ADDR_ORG. You must eliminate duplicate row IDs.
- You must evaluate whether to modify upgrade scripts to migrate address data in custom extension columns in S_ADDR_PER and S_ADDR_ORG. Review the new schema and determine whether your custom columns can be mapped to standard columns. If no matching standard columns exist, then create new columns on the target tables. During the database upgrep, you do this after running the Database Configuration Utilities but before running the Upgrade Wizard.
- After the upgrade is complete, review the records in S_ADDR_PER and eliminate duplicate and obsolete records.

To manage address migration, follow the steps in *Process of Upgrading a Siebel Production Test Environment*. Each of the address migration tasks is included as a step in this process. Each step refers you to a procedure for performing the task.

Upgrade Planning for Siebel Workflow Designer

Platforms: Windows, UNIX, IBM z/OS.

The repository merge process preserves customizations you have made to seeded workflows. A seeded workflow is a Workflow Process object and all child objects shipped as part of a Siebel release.

Special premerge and postmerge steps have been added to the regular repository merge to allow merging customer-modified seeded workflows:

1. **Workflow premerge.** When you start a repository merge, this step runs and prepares customer-modified seeded workflows in the Prior Customer Repository for the repository merge.
2. **Repository merge.** After the workflow premerge completes, the main repository merge runs. The repository merge identifies modifications you have made to seeded workflows in the Prior Customer Repository. It then copies the modifications to the corresponding seeded workflows in the New Customer Repository.
3. **Workflow postmerge.** After the main repository merge completes, the workflow postmerge runs and sets workflow versions in the New Customer Repository.

All three parts of the repository merge display in the status bar of the Merge Repositories dialog box when you perform a repository merge.

The workflow merge steps fit into the overall flow of a repository merge as follows:

- Repository merge starts
- Workflow premerge
- Main repository merge
- Workflow post-merge
- Merge ends
- Run the postmerge utilities

Workflow Premerge

When you start the repository merge, the workflow premerge step runs and prepares customer-modified seeded workflows for the repository merge.

If the workflow exists in both the Prior Customer Repository and New Siebel Repository, and the version number in the Prior Customer Repository is greater than 0, then the workflow is customer-modified.

For these workflows, the premerge process makes the following changes:

- Deletes version 0 of the customer-modified seeded workflow in the Prior Customer Repository.
- In the Prior Customer Repository, copies the most recent version with status Completed of the customer-modified seeded workflow and sets the copy’s version to 0.

  The most recent version of the workflow is the one with the highest version number and a status of Completed. This is called the version n workflow.

- Sets the copy's status to Completed.

  The workflow premerge creates a version 0 copy of version n in the Prior Customer Repository because the merge process requires that workflows it compares between repositories must have the same name, must be version 0, and must have a status of Completed.

Repository Merge

The main repository merge compares workflows in the Prior Customer Repository, Prior Siebel Repository, and the New Siebel Repository:

- **Customer-Modified seeded workflows.** The repository merge copies the modifications to version 0 of the workflow to the Prior Customer Repository to the same workflow in the New Customer Repository. The merge also copies versions 1 through n of the workflow to the New Customer Repository.

  An exception is when a workflow's object attributes are different in all three compared repositories. This means the object is both customer-modified and has also changed in the new release. This causes a merge conflict.

  The repository merge handles workflow attribute conflicts in the same manner as other objects. The merge process typically resolves merge conflicts in favor of the New Siebel Repository. You can review workflow-related attribute conflicts in the Application Upgrade Attribute List screen in Siebel Tools.

  If you modify a seeded workflow by deleting any of its child objects, then the merge process does not delete the child objects in the New Customer Repository. After the merge, you must review child objects and delete them as desired.
- **Unmodified seeded workflows.** If the workflow exists in both the Prior Customer Repository and New Siebel Repository, and the highest version level is 0 in the Prior Customer Repository, then the workflow is unmodified.

  After the merge, the seeded workflow in the New Customer Repository will be the one that shipped with the new release.
- **Customer-Created workflows.** If the workflow exists in the Prior Customer Repository but not the Prior Siebel Repository, then the workflow is customer-created. The repository merge copies all versions of customer-created workflows to the New Customer Repository.
- **Customer-Deleted Workflows.** If the workflow exists in the Prior Siebel Repository but not the Prior Customer Repository, then the workflow is customer-deleted. The repository merge does not delete the workflow from the New Customer Repository. After the merge, you must review these workflows and delete them as desired.
- **Obsolete seeded workflows.** If the workflow exists in the Prior Siebel Repository but not the New Siebel Repository, then the workflow is obsolete. The repository merge does not copy obsolete seeded workflows from the Prior Customer Repository to the New Customer Repository unless they are customer-modified. After the merge, you must review customer-modified, obsolete workflows and delete them as desired.

Workflow Postmerge

The workflow postmerge step runs after the main repository merge completes and does the following for seeded workflows that are customer-modified:

- Changes the version of the workflow from 0 to n + 1 and sets the n+1 version's workflow status to In Progress.

  For example, if the most recent version of customer-modified Workflow A in the Prior Customer Repository is version 3, then the merged version in the New Customer Repository will be version 4 and will have Status indicates In Progress.
- Copies version 0 of the workflow from the New Siebel Repository to the New Customer Repository. This reinstates the version 0 seeded workflow that shipped with the new release.

Status of Customer-Modified Workflows After the Merge

After the whole repository merge is complete, customer-modified seeded workflows appear as follows in the New Customer Repository:

- **Version 0.** This is the seeded workflow that shipped with the new release.
- **Versions 1 through n.** These versions are copied intact from the Prior Customer Repository.
- **Version n+1**. This is the new merged workflow version. It is a combination of the seeded workflow that shipped with the new release and the modifications contained in version n in the Prior Customer Repository.

Logging

Workflow premerge and postmerge. The workflow premerge and postmerge steps write to the same log file:

```
$SIEBEL_HOME\bin\merge0_ver.txt
```

Each time you run the merge process, the name of the `merge0_ver.txt` file is incremented, for example to `merge1_ver.txt`.

If the log file contains the error IDS_ERR_DEV_MRG_PREMERGE_FAILED, then this means that the premerge step could not delete one or more version 0 seeded workflows. This causes the main merge process to be unable to merge the customer-modified seeded workflow into the New Customer Repository. You must manually reimplement customizations to these workflows. You need not rerun the repository merge.

If the log file contains postmerge errors, then create a service request (SR) on My Oracle Support, or contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.

Main repository merge. The main repository merge logs workflow-related messages to the standard merge log file, merge0.txt. If you find workflow-related errors in this log file, then create a service request (SR) on My Oracle Support, or contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.

The main repository merge log file is located in the same directory as the log file for the workflow premerge and postmerge steps.

Monthly Updates 20.8 – 22.6 to 22.7+ - Workflow Process Consideration

Updating from release 20.8 or higher (20.8-22.6) to a Monthly Update that is 22.7 or higher, may cause duplicate record insert issues for Workflow Processes during workspace Delivery/Rebase. The Delivery/Rebase may fail or cause a crash.

In Siebel Tools/Web Tools, Workflow Processes having multiple versions have the same Process Name such as "My Workflow Process", but the actual names at database level are "My Workflow Process:1", "My Workflow Process:2", … "My Workflow Process:6" in NAME column. This is expected and correct for pre-IP2017 versions.

With the introduction of Workspaces, Workflow versions are now tracked in the WS_OBJ_VER column and hence every Workflow record must have a unique Name to avoid duplicate insert issues.

How the application identifies the one true definition of a Workflow Process

If this is a Workflow Process present in the Repository we shipped with the product and found in the MAIN Workspace, it is the "original" definition of that Workflow Process. Its Row Id is the logical pointer that all Workspaces modifying that Workflow Process will use in their WS_SRC_ID column. The Workspaces all point back to the definition in MAIN using that Row Id. Every Workspace created directly under MAIN will point to the Workflow Process definition found in MAIN. During a merge or rebase, the definition in MAIN will be used. Oracle may modify the Workflow Process definition at some point and ship it with a Monthly Update through the RepositoryUpgrade utility. In this case, the definition we are shipping will import into the Repository using the same Row Id as the original version we shipped. As it is delivered up the Workspace chain, the "original" version should still point to the one in MAIN although the definition may have been changed by you previously.

Creating a different copy of the Workflow Process through a SIF import

Sometimes, developers export a Workflow Process to a Siebel Archive File (SIF) and import it into a different Workspace. When that happens, the WS_SRC_ID does not point to the one in MAIN. It has a new WS_SRC_ID created in that Workspace. Now the situation is that there are two Workflow Processes with the same name but different WS_SRC_IDs. Now the application will not be able to know which one should be used This is one of the sources of the issues that this utility fixes.

There will only be one logical pointer to an object even though this object may be modified in many different Workspaces simultaneously. When a Workspace is delivered, the application must merge the changes made in the Workspace with other delivered versions of this object. For instance, there is only one SIS OM Active Order Sub-Process Workflow Process although it may have been modified in many different Workspaces. When a developer is modifying the SIS OM Active Order Sub-Process Workflow Process in a development Workspace, the WS_SRC_ID holds a logical pointer to the SIS OM Active Order Sub-Process Workflow Process found in the MAIN Workspace that is considered the Golden Version (the source of truth). All the developer Workspaces, if modifying the SIS OM Active Order Sub-Process Workflow Process should have the Row Id of the SIS OM Active Order Sub-Process Workflow Process as it is found in the MAIN Workspace. Sometimes a Repository may have records in a Workspace that do not have the expected, logical pointer to the SIS OM Active Order Sub-Process Workflow Process in the WS_SRC_ID column. In such a case the deliver log may show errors such as the following.

Example of delivery errors

```
LINE 16610: DUPLICATE OBJECTS FOR TYPE 'Workflow Process' AND PATH 'Configurator Product Info Lookup'
LINE 17384: DUPLICATE OBJECTS FOR TYPE 'Workflow Process' AND PATH 'ISS Promotion WS - Add Missed Items Sub
 Process'
LINE 17690: DUPLICATE OBJECTS FOR TYPE 'Workflow Process' AND PATH 'SIS OM Active Order Sub-Process'
```

Remediation

In the update scenario described previously, run a new utility named **WFCleanup.exe** after you finish the Monthly Update. This utility is found in the `<SIEBEL ROOT>/BIN`.

Running the Utility

```
WFCleanup.exe -t <Table Owner> -u <Table Owner User Name> -p <Table Owner Password> -o
< ODBC Data Source > -r <Repository Name> -s <SIEBEL ROOT/BIN> -d <DB Name>
```

Examples of running the Utility

Microsoft SQL Server

```
WFCleanup.exe -t dbo -u MSNMK122 -p ******** -o MSNMK122 -r "Siebel Repository" -s
"C:\Siebel\22_10" -d MSSQL
```

Oracle

```
WFCleanup.exe -t ORAUJD110 -u ORAUJD110 -p ******** -o ORAUJD110 -r "Siebel Repository" -s
"C:\Siebel\22_10" -d ORACLE
```

DB2UDB

```
WFCleanup.exe -t SIEBEL -u SIEBEL -p ******** -o DB2DM115 -r "Siebel Repository" -s
"C:\Siebel\22_10" -d DB2UDB
```

DB2390

```
WFCleanup.exe -t CQ10C002 -u QADMIN -p ******** -o Q10C -r "Siebel Repository" -s
"C:\Siebel\22_10" -d DB2390
```

Argument List for the Utility

Argument	Description	Comment
-t	Siebel Table Owner	Required
-u	TBLO Username	Required
-p	TBLO Password	Required
-o	ODBC Data Source	Required
-r	Repository Name	Default: "Siebel Repository"
-s	Siebsrvr/Tools Installation path specified	Required
-d	DB Platform Name	Oracle, MSSQL,DB2UDB or DB2390

Utility Output Log Location

The utility Logs are generated in `<SIEBEL SERVER_ROOT>\log`.

The WFCleanup.exe performs the following actions:

1. Identifies the primary record for each Workflow having identical process names but different WS_SRC_IDS. The process with the highest version that is in the status of COMPLETED is treated as the PRIMARY record. For this primary record, it updates the NAME value to what is in the PROC_NAME column. For the other records with the same name the PROC_NAME column will be updated with NAME value and the record is inactivated by setting its INACTIVE_FLG = Y. This makes each record unique.

   Note: If highest version of the Workflow in Completed status is also inactive, then it makes highest versioned IN_PROGRESS record the surviving Workflow record.

   Example of records before running the utility

NAME	PROC_NAME	VERSION	STATUS_CD	INACTIVE_FLG
ABO Bulk Request - Validate Process:0	ABO Bulk Request - Validate Process	0	Completed	N
ABO Bulk Request - Validate Process:1	ABO Bulk Request - Validate Process	1	In Progress	N
ABO Bulk Request - Validate Process:2	ABO Bulk Request - Validate Process	2	Completed	N
ABO Bulk Request - Validate Process:3	ABO Bulk Request - Validate Process	3	In Progress	N

   Example of records after running the utility

NAME	PROC_NAME	VERSION	STATUS_CD	INACTIVE_FLG
ABO Bulk Request - Validate Process:0	ABO Bulk Request - Validate Process:0	0	Completed	Y
ABO Bulk Request - Validate Process:1	ABO Bulk Request - Validate Process:1	1	In Progress	Y
ABO Bulk Request - Validate Process	ABO Bulk Request - Validate Process	2	Completed	N
ABO Bulk Request - Validate Process:3	ABO Bulk Request - Validate Process:3	3	In Progress	Y

Note: The surviving Workflow record in this case is version 2. It was in a status of Completed and had the highest version. The NAME of this record changes from ABO Bulk Request -Validate Process:2 to ABO Bulk Request -Validate Process and becomes the only active record for the Runtime Repository once migrated. The PROC_NAME is updated with the value in the NAME column and the INACTIVE_FLG is set to Y for the remaining records.

2. Workflow records with the same name but with different versions, all with a status of "In Progress"are handled in the following manner. The most recent IN PROGRESS record is considered as primary. This primary workflow NAME is then renamed with PROC_NAME and Activated. For rest of the records PROC_NAME is changed to NAME to make them unique, and Inactivated.

Example of records before running the utility

NAME	PROC_NAME	VERSION	STATUS_CD	INACTIVE_FLG
ABO Bulk Request - ApplyProductPromotion:0	ABO Bulk Request - ApplyProductPromotion	0	In Progress	N
ABO Bulk Request - ApplyProductPromotion:1	ABO Bulk Request - ApplyProductPromotion	1	In Progress	N

Example of records after running the utility

NAME	PROC_NAME	VERSION	STATUS_CD	INACTIVE_FLG
ABO Bulk Request - ApplyProductPromotion:0	ABO Bulk Request – ApplyProductPromotion:0	0	In Progress	Y
ABO Bulk Request - ApplyProductPromotion	ABO Bulk Request - ApplyProductPromotion	1	In Progress	N

Note: The latest Version is 1. The NAME of this record changes from ABO Bulk Request - ApplyProductPromotion:1 to ABO Bulk Request – ApplyProductPromotion. The other In Progress record has been inactivated and the PROC_NAME column has been updated with the value in the NAME column.

Upgrade Planning for Mobile Devices in the Siebel Environment

Platforms: Windows, UNIX, IBM z/OS.

Before the upgrade, verify that mobile devices are running an operating system supported by your version of Siebel CRM. Also, verify that third-party software is the correct version. For more information, see the Certifications tab on My Oracle Support.

After the upgrade, you must enter any mobile device-related application configuration changes into the mobile device administration screen.

Mobile Application Upgrade

Mobile applications do not upgrade automatically. Users might need to download the latest version of the application after an upgrade to the current release. It is not necessary to uninstall the application first.

Upgrade Planning for Resonate Central Dispatch in the Siebel Environment

Platforms: Windows, UNIX, IBM z/OS.

Support for Resonate Central Dispatch is discontinued. It has been replaced by a load balancing module that is included in the Siebel Web Server Extension and its successor module, Siebel Application Interface. For a description of the load balancing module and information about requirements for third-party HTTP load balancers, see *Siebel Deployment Planning Guide* . For information on configuring load balancing, see *Siebel Installation Guide* .

Upgrade Planning for Siebel String Translation

Platforms: Windows, UNIX, IBM z/OS.

Based on the language for the Siebel upgrade process, the Siebel upgrade scripts are created accordingly.

> **Note:** To avoid unintended results, you must perform your upgrade in the same language as that of the base language in the prior release.

When the Database Configuration Wizard is started, the sqlgen utility generates the Siebel upgrade scripts based on the information located in the upgfile.xml file. When the Database Configuration Wizard completes, SQL scripts are generated, and replace the placeholders in the upgfile.xml with the language-specific values located in the `DBSRVR_ROOT/lang_code/upglocale.lang_code` file.

In the following example, the locale file is upglocale.fra, for a FRA based installation. The upgfile.xml contains the following statement:

```
update S_DOC_QUOTE
```

Set the QUOTE_SUB_TYPE_CD value to `&apos;<XTL_STRING>Private</XTL_STRING>&apos;`

Where the QUOTE_TYPE equals `&apos;<XTL_STRING>Template</XTL_STRING>&apos;`

The placeholders `<XTL_STRING>Private</XTL_STRING>` and `<XTL_STRING>Template</XTL_STRING>` are values that the sqlgen utility changes during the upgrade. The XTL_STRING values are replaced based on what the sqlgen utility locates as a match for the Private and Template strings in the upglocale.fra file, located under `dbsrvr/FRA`. For this example, the FRA base language dictates Template = Modèle and Private = Privé.

CAUTION: If you modified string mappings in your previous Siebel release, then you must contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services in order to modify standard upgrade scripts accordingly. If you do not, then your Siebel application data from the prior release might not be migrated to the new Siebel release's data model.

Upgrade Planning for Siebel Personalization

Access control is based on primary responsibility name. If you have defined conditional expressions for applets in the Administration - Personalization screen, then plan to review these after the upgrade and verify that they use Primary Responsibility name. For more information on access control, see *Siebel Security Guide* . For more information on personalization management, see Siebel Personalization Administration Guide.

Upgrade Planning for Siebel Pricer and Order Management

Data associated with Siebel Pricer features are upgraded as shown in the following table. These upgrade changes apply to upgrades from any Siebel CRM version 7.x release.

Feature	How This Feature Upgraded
Price lists and cost lists	Automatically upgraded as part of price list
Customizable product pricing	Automatically upgraded as part of price list
Service pricing (based on covered product)	Automatically upgraded as part of price list
Volume discount	Automatically upgraded
Attribute pricing	Automatically upgraded as attribute adjustment
Pricing model • Aggregate factor • Bundle factor • Single factor • Matrix factor • Script based	Not automatically upgraded. Must be redesigned and reimplemented as pricing procedures. Bundle factor definitions are upgraded to aggregate discounts and aggregate discount sequences.

Siebel Pricer API user properties for internal or external application integration are also obsolete. After the upgrade, you must reimplement integrations using pricing procedures, signals, variable maps and other features of the order management infrastructure.

For information on the new Pricer architecture, on application integration, and on order management infrastructure, see *Siebel Pricing Administration Guide* and Siebel Order Management Infrastructure Guide.

For a scenario that describes how to reimplement Siebel CRM version 7.x pricing models, see 473908.1 (Article ID) on My Oracle Support. This document was previously published as Siebel Technical Note 639.

5 Preparing for Siebel Database Upgrade

Preparing for Siebel Database Upgrade

This chapter provides initial preparatory information for the Siebel database upgrade. This chapter includes the following topics:

Verifying Siebel Database Connectivity

Environments: Development, production test, production.

From the production test environment, you must be able to make ODBC connections to both the Siebel database in the development environment and the Siebel database in the production environment. Verify that you can define these ODBC connections in the production test environment.

If you cannot connect to these databases from the production test environment, then create a service request (SR) on My Oracle Support, or contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.

In the production environment, you do not have to define an ODBC connection to the development environment Siebel database or the production test environment Siebel database.

Preparing Siebel Tables and Views for Upgrade

Environments: Development, production test, production.

Platforms: Windows, UNIX, IBM z/OS.

To prepare tables and views for upgrade

1. Remove temporary tables and non-Siebel tables.

If the upgrade process detects a column with a datatype not acceptable for Siebel tables, then the upgrade will fail.

2. Disable customized triggers.

   You must re-create them after the upgrade.
3. Remove defined database views on Siebel tables.

   You must re-create them after the upgrade.
4. Export interface table data that you want to preserve.

   Interface tables are removed and then re-created during upgrade. You can import the data after the upgrade.

Preparing Siebel Custom Indexes for Upgrade

Environments: Development, production test, production.

Platforms: Windows, UNIX, IBM z/OS.

Consider the following guidelines when preparing custom indexes for upgrade:

- **Custom indexes against extension columns on obsolete tables.** If you have created custom indexes that use extension columns on obsolete tables, then you must migrate the data to new extension columns before upgrading the Siebel database. For assistance, create a service request (SR) on My Oracle Support, or contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.
- **Custom indexes that were not defined through Siebel Tools.** Custom indexes created without using Siebel Tools are not included in the schema definition in the Siebel Repository. These indexes are dropped during the database upgrade. To preserve these indexes, add them to the Siebel Repository using Siebel Tools.
- **Custom indexes on interface tables.** Custom indexes on interface tables are not re-created during the upgrade. You must re-create them after the upgrade is complete.
- **Custom indexes on base tables.** The upgrade automatically removes and re-creates custom indexes on base tables.
- **Custom indexes might need to be changed to reflect schema changes.** Reevaluate custom indexes for applicability in the new release. They might no longer be needed due to schema changes in the new release.

For more information about custom indexes, see *Configuring Siebel Business Applications* . For information on schema changes in a release, see *Siebel Data Model Reference* .

Exporting Siebel Interface Table Data

Environments: Development, production test, production.

Platforms: Windows, UNIX, IBM z/OS.

During the upgrade process, your interface tables are dropped and then re-created. To retain data in your interface tables, use the appropriate tools for your RDBMS to export data before the upgrade and then import the data after you have completed the upgrade.

During the upgrade, all custom indexes on interface tables are dropped from both logical and physical schema.

Archiving Unneeded Siebel Repositories

Environments: Development, production test, production.

Perform this task before running the Database Configuration Utilities in upgrep mode for the first time in an environment.

The upgrep mode scripts expect only the Siebel Repository to be present. If you have multiple repositories, then you must export them to archive files and then delete them from the database.

Because the upgrade changes the schema of the database, in most cases you cannot import these archived repositories into the upgraded database. If you want to access the archived repositories, then you must import them into a database that has the same schema as the one from which they were exported.

To archive unneeded repositories

1. Export all repositories.
2. Place exported repository files in a safe location.
3. In the Siebel database, delete all repositories except the Siebel Repository.

For information on exporting and deleting repositories, see *Using Siebel Tools* .

Preserving Siebel Dock Objects and Visibility Rules

Environments: Development, production test, production.

Platforms: Windows, UNIX, IBM z/OS.

Modified visibility rules are dropped during a development environment upgrade. Manually record any changes to dock object visibility rules, so you can evaluate whether you must reapply the changes after the upgrade is complete.

Dock objects and visibility rules created by using Docking Wizard are preserved unless they become invalid after the upgrade. Manually record any changes that you made through the Docking Wizard so that you can evaluate whether you must reapply the changes after the upgrade is complete.

Changing the definition of dock objects requires the assistance of Oracle Advanced Customer Services. Contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.

Securing AIX Memory Allocation Segment Space for the Siebel Database

Environments: Development, production test, production.

Databases: All databases.(Exception: this topic does not apply to Microsoft SQL Server.)

Platforms: UNIX only.

Before you run an upgrade on AIX, set the following environment variable on the AIX computer that you are using for the upgrade:

```
setenv LDR_CNTRL LOADPUBLIC@MAXDATA=0x60000000
```

This will prevent a shortage of memory allocation segment space that might occur on the computer where both the Siebel Database Server and Siebel Server are installed. After a successful upgrade, reset this parameter to the original value.

Preparing for a Multilingual Upgrade

Environments: Development, production test, production.

Platforms: Windows, UNIX, IBM z/OS.

The upgrade process upgrades both the primary (base) language and all deployed languages.

The Database Configuration Utilities validate deployed languages by comparing the status of the language packs installed on the Siebel File System with the language IDs of records in the S_LST_OF_VAL table. If there are language IDs in S_LST_OF_VAL but no corresponding deployed language on the Siebel File System, then the validation fails and you cannot continue the upgrade.

Before doing an upgrep, use Siebel Tools to remove or deactivate any records in S_LST_OF_VAL for undeployed languages.

To prepare for a multilingual upgrade

1. Make a list of your deployed languages.

   A *deployed language* is one where you have installed the language pack, imported the user interface strings into the Siebel Repository, imported the seed data into the database, activated the seed data records added to S_LST_OF_VAL, and configured the records for MLOVs.
2. In Siebel Tools, navigate to Screens, System Administration, and then List of Values.
3. Locate the Language Name field, and verify that no languages appear that are not currently deployed.

   Even if you do not have English installed, then some records will have the Language Name value equal English-American. You can ignore these system records.

4. For records where you do not have the corresponding language pack deployed, investigate why these records are present and take one of the following actions:

   - If the record is not needed, delete it.
   - If you are not sure whether the record is needed, remove the check-mark from the Active field. This deactivates the record and prevents it from being included in the validation check performed by the Database Configuration Utilities.

     > **CAUTION:** Do not deactivate or delete records where the Language Name equals English-American, even if you do not have English-American deployed. These records are needed by the system. The validation process ignores these records.

6 Preparing an IBM DB2 Database for a Siebel Upgrade

Preparing an IBM DB2 Database for a Siebel Upgrade

This chapter provides initial preparatory information for a Siebel database upgrade on IBM DB2. This chapter includes the following topics:

- *Verifying the IBM DB2 Client for a Siebel Upgrade*
- *Verifying IBM DB2 Sort Order for a Siebel Upgrade*
- *Setting IBM DB2 Parameters for a Siebel Upgrade*
- *Verifying IBM DB2 Permissions for a Siebel Upgrade*
- *Verifying IBM DB2 Instance Owner Permissions for a Siebel Upgrade*
- *Creating IBM DB2 Temporary Tablespaces and Bufferpools for a Siebel Upgrade*
- *Analyzing IBM DB2 Custom Tablespace Requirements for a Siebel Upgrade*
- *Verifying the IBM DB2 Application Development Client for a Siebel Upgrade*
- *Identifying IBM DB2 Long Columns for Truncation in a Siebel Upgrade*

Verifying the IBM DB2 Client for a Siebel Upgrade

Environments: Development, production test, production.

Databases: IBM DB2 UDB only.

The Siebel Server supports only the 32-bit IBM DB2 client. Verify that you have not installed the 64-bit IBM DB2 client on the Siebel Servers. If you have installed the 64-bit IBM DB2 client, then replace it with the 32-bit client.

IBM supports the 32-bit IBM DB2 client working with 64-bit IBM DB2 databases.

Verifying IBM DB2 Sort Order for a Siebel Upgrade

Environments: Development, production test, production.

Databases: IBM DB2 only.

Binary sort order is required for the development environment upgrade and is strongly recommended for the production environment upgrades.

Sort order is specified during creation of the database. If you find that your IBM DB2 development database was not created using Identity sort order, then you must re-create your database using the option `COLLATE USING IDENTITY`.

If sort order is correct, but you are still encountering errors, contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services to help in further analysis.

To verify that your database was created using Identity sort order

1. Run the following query in the Siebel database:

   ```
   select count (*) from SIEBEL.S_APP_VER where '$' > '/'
   ```

2. Review the result.

   - If sort order is correct, then the result is as follows:

     ```
     1

     --------------

     0

     (1) record selected.
     ```

   - If sort order is incorrect, then you must re-create the database, using this option:

     ```
     COLLATE USING IDENTITY
     ```

Setting IBM DB2 Parameters for a Siebel Upgrade

Environments: Development, production test, production.

Databases: IBM DB2 only.

Before upgrading an IBM DB2 database, verify that your database server meets or exceeds the following configuration criteria:

- The DMS tablespace has at least 25% of free pages.
- The file system has sufficient space to allow your DMS tablespace to grow.
- Siebel tablespaces for IBM DB2 must be database-managed tablespaces (DMS) rather than system-managed tablespaces (SMS).
- Verify that the tablespaces are not near their capacity. This can be done by connecting to the database and issuing the following command:

  ```
  DB2 list tablespaces show detail
  ```

The tables which follow provide upgrade-specific settings for the Database Manager and database. Use the following strategy to set parameters:

- Set parameters using the recommendations in *Siebel Installation Guide* . Recommendations are located in the chapter on configuring the RDBMS.
- For the upgrade, revise the configuration parameters listed in the tables which follow.
- After the upgrade, reset the configuration parameters to the values listed in *Siebel Installation Guide* .

IBM DB2 Database Manager Settings

The Upgrade Setting column in the following table provides guidelines for setting configuration parameters specifically to optimize upgrade performance. Set these parameters for each IBM DB2 instance.

Parameter	Explanation	Upgrade Setting
`SHEAPTHRES`	Sort heap threshold (4 KB) If you reset `SHEAPTHRES` or `SORTHEAP`, then rebinding the instance is recommended.	Double the value allocated for `SORTHEAP`. See the following table.

IBM DB2 Database Configuration Parameters

The Upgrade Setting column in the following table provides guidelines for setting configuration parameters specifically to optimize upgrade performance. Set these parameters for each IBM DB2 instance.

IBM DB2 Database Configuration Parameters

Parameter	Explanation	Upgrade Setting
`SORTHEAP`	Sort list heap (4 KB)	`20000—40000` Recommended size; this might increase or decrease depending on the amount of memory in the database server computer and the size of the data. A `20000` setting allows `SORTHEAP` to increase up to 80 MB.
`MAXLOCKS`	Percentage of lock lists for each application	`5`
`CHNGPGS_THRESH`	Changed pages threshold	`5`
logarchmeth1	Primary log archive method configuration parameter	OFF. To retain active log files for rollforward recovery, set logarchmeth1 to LOGRETAIN by issuing the command: `UPDATE DB CFG USING logarchmeth1 LOGRETAIN.`
`LOGFILSIZ`	Log file size (4 KB)	Development environments: `8000-16000`
`SOFTMAX`	Triggers bufferpool flushing	`50`

Verifying IBM DB2 Permissions for a Siebel Upgrade

Environments: Development, production test, production.

Databases: IBM DB2 only.

Platforms: UNIX only.

If you are running IBM DB2 on IBM AIX or Oracle Solaris, then perform the following steps before executing the Siebel Database upgrade.

To verify IBM DB2 permissions

1. Navigate to the instance home directory.
2. Use the following command to verify that the directory `sqllib/function/routine/sqlproc` has write permission for the group:

   ```
   ls -ld sqllib/function/routine/sqlproc
   ```

3. To authorize group write permission, enter the following command:

   ```
   chmod g+w sqllib/function/routine/sqlproc
   ```

Verifying IBM DB2 Instance Owner Permissions for a Siebel Upgrade

Environments: Development, production test, production.

Databases: IBM DB2 only.

Platforms: UNIX only.

If you are running IBM DB2 on AIX or Oracle Solaris, then verify that the Siebel Database instance owner belongs to the primary group of the fenced user. If the instance owner is not part of this group, then errors will occur during the Siebel Database upgrade.

Creating IBM DB2 Temporary Tablespaces and Bufferpools for a Siebel Upgrade

Environments: Development, production test, production.

Databases: IBM DB2 only.

If your RDBMS is IBM DB2, then verify that you have 16-KB and 32-KB temporary tablespaces to use for sorting and other SQL processing. Both the 16-KB and 32-KB temporary tablespaces require dedicated bufferpools.

Creating a 16-KB Temporary Tablespace

Use the following procedure to create a 16-KB temporary tablespace.

To create a 16-KB temporary tablespace

1. Create a 16-KB bufferpool with at least 5000 16-KB pages.
2. Create a 16-KB temporary tablespace as system managed space (SMS) that can be expanded to 2 GB of storage.

Creating a 32-KB Temporary Tablespace

Use the following procedure to create a 32-KB temporary tablespace.

To create a 32-KB temporary tablespace

1. Create a 32-KB bufferpool with at least 1000 32-KB pages.
2. Create a 32-KB temporary tablespace as SMS that can be expanded to 2 GB of storage.

Analyzing IBM DB2 Custom Tablespace Requirements for a Siebel Upgrade

Environments: Development, production test, production.

Databases: IBM DB2 only.

There are four standard database managed tablespaces (DMS) that hold Siebel tables and indexes: a 4-KB, 8-KB, 16-KB, and 32-KB tablespace for tables, and a tablespace to hold indexes. The upgrade process moves tables between these spaces as required.

If you have placed Siebel tables in other tablespaces, then the upgrade process will not move these tables if they grow to exceed the tablespace size during the upgrade. If one of these tables has an estimated page size after upgrade greater than its current page size, then it will not fit in its tablespace after the upgrade, and the upgrade will fail.

Oracle provides a sizing utility that determines whether tables will increase in size to the point that they must be moved to a larger tablespace.

Run the utility before upgrading the database. If the sizing utility reports any problems, then you must resolve them before you proceed with the upgrade.

To analyze tablespace requirements for IBM DB2

1. Navigate to the following directory:

   Windows: `SIEBEL_ROOT\bin`

UNIX: `$SIEBEL_ROOT/bin`

2. Type the following command line:

```
tblsize /U TABLEOWNER /P PASSWORD /C ODBC_DATASOURCE /F DDL_FILE /B DEFAULT_TABLESPACE
/X DEFAULT_INDEXSPACE /K 16K_TABLESPACE /V 32K_TABLESPACE /Q REPORT_FILENAME
/L LOG_FILENAME /Z UCS2_DATABASE /A DEBUGMODE
```

   where:

   - `TABLEOWNER` is the tableowner.
   - `PASSWORD` is the tableowner password.
   - `ODBC_DATASOURCE` is the data source of the database.
   - `DDL_FILE` is the absolute path to the DDL file (this file is called `ddl.ctl`, and it is located in the `dbsrvr/DB2` directory).
   - `DEFAULT_TABLESPACE` is the name of the 4-KB page standard Siebel tablespace.
   - `DEFAULT_INDEXSPACE` is the name of the standard Siebel index space.
   - `16K_TABLESPACE` is the name of the 16-KB page standard Siebel tablespace.
   - `32K_TABLESPACE` is the name of the 32-KB page standard Siebel tablespace.
   - `REPORT_FILENAME` is the name of the report generated by the utility.
   - `LOG_FILENAME` is the name of the log file (default: `custtbl.log`).
   - `UCS2_DATABASE` specifies whether the database uses Unicode or Non-Unicode (default: N).
   - `DEBUGMODE` retrieves the logs in detail (default: N).

   Example:

```
tblsize /U siebel /P siebel /C ssia /F d:\sea77\dbsrvr\DB2\ddl.ctl /B siebel_4k
/X siebel_idx /K siebel_16k /V siebel_32k /Q d:\sba82\dbsrvr\DB2\report.txt
/L $SIEBEL_ROOT/log/tblsize.log /Z Y /A Y
```

3. Review the report generated by the utility to determine whether the estimated table pagesize postupgrade is larger than the size of the actual custom table pagesize.

   An example of the report generated by this utility is displayed in the following example:

```
 Table Name = S_EVT_ACT
Custom Tablespace Id = 5
Custom Tablespace Name = CUST_TBS_EVT_ACT
Custom Tablespace Pagesize = 4096
Estimated Table Pagesize (postupgrade) = 5067
Status = Does not fit in its custom tablespace
```

CAUTION: For each table that has `Status: Does not fit in its custom tablespace`, you must create a larger custom tablespace that is larger than the estimated table pagesize postupgrade.

4. Move the tables from their old tablespaces to the new ones by running `ddlmove`.

   `ddlmove` is a utility for moving tables from one tablespace to another tablespace. This utility is located in the following directory:

   Windows: `SIEBEL_ROOT\bin`

   UNIX: `$SIEBEL_ROOT/bin`
5. To run `ddlmove`, submit the following arguments:

```
ddlmove /U TABLEOWNER /P TABLE_PASSWORD /C ODBC_DATASOURCE /E STOP_ON_DDL_ERROR
/G GRANTEE /B TABLESPACE /X INDEX_TABLESPACE /M TABLE_NAME
/L LOG_FILENAME/Z UCS2_DATABASE
```

   where:

   - `TABLEOWNER` is the tableowner of the database (required).
   - `TABLE_PASSWORD` is the password of the tableowner of the database (required).
   - `ODBC_DATASOURCE` is the data source of the database (default environment variable: SIEBEL_DATA_SOURCE).
   - `STOP_ON_DDL_ERROR` is the stop on DDL Error parameter (default: Y).
   - `GRANTEE` is the grantee for tables (SSE_ROLE).
   - `TABLESPACE` is the name of the tablespace that you are moving the table to.
   - `INDEX_TABLESPACE` is the name of the index space that you are moving the table to.
   - `TABLE_NAME` is the Table Name Like Support value (default: N).
   - `LOG_FILENAME` is the name of the log file (default: ddlmove.log).
   - `UCS2_DATABASE` specifies whether the database uses Unicode (default: N).

Verifying the IBM DB2 Application Development Client for a Siebel Upgrade

Environments: Development, production test, production.

Databases: IBM DB2 only.

The IBM DB2 Application Development Client must be installed on the RDBMS server. The following table lists the required IBM DB2 Application Development Client components.

Operating System	IBM DB2 Application Development Client Components
Microsoft Windows	DB2 Application Development Client
IBM AIX	Application Development Tools (ADT) ADT Sample Programs

Operating System	IBM DB2 Application Development Client Components
HP-UX	Application Development Tools for HP-UX
Oracle Solaris	Application Development Tools (ADT) ADT Sample Programs

For information on installing the Application Development Client, see IBM documentation.

Identifying IBM DB2 Long Columns for Truncation in a Siebel Upgrade

Environments: Development, production test, production.

Databases: IBM DB2 only.

In Siebel CRM version 7.7, the maximum length for IBM DB2 long columns with a type of varchar was reduced to 16,350 from 16,383. Upgrading from version 7.5.3 truncates long varchar columns that exceed 16,350. To prevent a data truncation error that might cause transaction processing (`txnproc`) or transaction routing (`txnroute`) to fail, perform the steps in this task to identify these columns and reduce the data in these columns.

CAUTION: If you do not truncate the data in long varchar columns that exceed the maximum length specified in the following task, then a `data truncated` error occurs, and transaction processing and transaction routing might fail.

To identify and reduce the length of long varchar columns

1. From any shell, open the script `chk16350.bat` (Windows) or `chk16350.ksh` (UNIX), and edit the following parameters as appropriate for your deployment:

    - `SRC_USR` is the username of the source database
    - `SRC_PSWD` is the password for the source database
    - `SRC_TBLO` is the tableowner of the source database
    - `SRC_TBLO_PSWD` is the tableowner password for the source database
    - `SRC_ODBC` is the ODBC data source name of the source database (edit the value `CHANGE_ME`)
    - `SRC_REPOSITORY_NAME` is the repository name of the source database
    - `DBSRVR_ROOT` is the directory where you installed the Siebel Database Server files on the Siebel Server, for example, `C:\sba81\dbsrvr` (Windows). Edit the value `CHANGE_ME`.
    - `SIEBEL_ROOT` is the directory where you installed the Siebel Server. For example, `C:\sba81\siebsrvr` (Windows). Edit the value `CHANGE_ME`.
    - `VALID_RESULTS_DIR` is the directory where you want the output files to be generated (edit the value `CHANGE_ME`); this must be an existing directory

    This script produces two files:

- **long_trunc_cols.rpt.** This report identifies all long varchar columns that are longer than 16,350 characters.
- **update_trunc.sql.** This SQL file generates update statements that truncate identified columns to 16,350 characters.

2. Reduce the data in these columns using either of the following methods:

    - Manually review the columns in the `long_trunc_cols.rpt` report and manually reduce the size of each column identified.
    - Run `update_trunc.sql` using the IBM DB2 command line processor.

7 Preparing an Oracle Database for a Siebel Upgrade

Preparing an Oracle Database for a Siebel Upgrade

This chapter provides initial preparatory information for a Siebel database upgrade on the Oracle Database. This chapter includes the following topics:

- *Verifying Oracle Database Sort Order for a Siebel Upgrade*
- *Setting Oracle Database Configuration Parameters for a Siebel Upgrade*
- *Verifying Oracle Database Parameters for Multiple Processors in a Siebel Upgrade*
- *Verifying the Oracle Database ODBC Definition for a Siebel Upgrade*

Verifying Oracle Database Sort Order for a Siebel Upgrade

Environments: Development, production test, production.

Databases: Oracle only.

The NLS_SORT parameter determines the sort order of query returns. NLS_SORT must be set to BINARY for the development environment upgrade, including the repository merge. It must also be set to BINARY when using Siebel Tools to publish the Siebel Runtime Repository.

This setting is strongly recommended for production test environment upgrades and production upgrades.

To verify that your database is using binary sort order

1. Run SQL*Plus to connect to the Oracle Database.
2. Issue the following query:

```
SQL> SELECT * FROM NLS_DATABASE_PARAMETERS;
```

3. In the returned parameters, locate `NLS_SORT` and verify that its value is `BINARY`.
   - If `NLS_SORT` has a value of `BINARY`, then the sort order is binary and no action is required.
   - If `NLS_SORT` is anything other than `BINARY`, then reset the value to `BINARY`.

Setting Oracle Database Configuration Parameters for a Siebel Upgrade

Environments: Development, production test, production.

Databases: Oracle only.

This topic provides upgrade-specific settings for the Oracle RDBMS. Use the following strategy to set parameters:

- For information about supported Oracle Database versions and any required patches, see the Certifications tab on My Oracle Support. If required, upgrade your Oracle Database and client software to the supported version, using Oracle's tools and documented procedures.
- Set parameters using the recommendations in 781927.1 (Article ID) on My Oracle Support. (This article was previously published at Siebel Technical Note 582.)
- Set additional parameters using the recommendations in *Siebel Installation Guide* for the operating system you are using. Recommendations are located in the chapter on configuring the RDBMS.
- Configure the database and set parameters as indicated in the following topics.
- After the upgrade, reset the configuration parameters to the values listed in *Siebel Installation Guide* for the operating system you are using and in 781927.1 (Article ID) on My Oracle Support. This document was previously published as Siebel Technical Note 582.

General Configuration

Before upgrading an Oracle Database, complete the following tasks:

- Verify that your Oracle clients and Oracle Database server meet the specifications certified for this release.
- **UNIX only**. If you have installed the Oracle 64-bit client on the Siebel Server, then verify that `$ORACLE_HOME/lib32` instead of `$ORACLE_HOME/lib` is included in LIBPATH (AIX), SHLB_PATH (HP-UX), LD_LIBRARY_PATH (Oracle Solaris).
- **pctincrease.** For upgrades, compute a high enough value for `pctincrease` on tablespaces that contain application tables and indexes so that upgrading does not create large numbers of extents.
- **pctfree.** Rebuild some of your larger tables with a large value for `pctfree` (30 or higher). Table size depends on which Siebel Business Applications you have deployed. For example, if you are upgrading Siebel Financial Services, then S_ASSET is a large table and S_ADDR_ORG is not used.

  You must increase `pctfree` before the upgrade because many new columns are added to tables during the upgrade. Migrating data into the new columns during the upgrade is likely to cause row chaining, which degrades upgrade performance.
- **DB_CACHE_SIZE.** Set this parameter to a minimum of 394264576.
- **SORT_AREA_SIZE.** Set this parameter to a minimum of 1524288. This significantly reduces the time required to complete a repository merge.

  See *Siebel Performance Tuning Guide* for other parameter settings.
- **UNDO_MANAGEMENT (Oracle Database 10*g* and later).** Set the UNDO_MANAGEMENT parameter to MANUAL before the repository merge. This turns off Automatic Undo Management (AUM). You can turn

AUM back on after the repository merge, as desired. For more information on how AUM affects upgrade, see 477025.1 (Article ID) on My Oracle Support. This document was previously published as Siebel Alert 848.

- **Rollback Segments.** Verify that you have only one large rollback segment on line that is appropriately sized so that the largest of transactions can be accommodated. Take all other rollback segments off line.

The upgrade might affect some of the largest tables in your implementation of Siebel CRM version 7.x, causing them to grow by as much as 40%.

Customer experience has shown that repository merges involving multiple languages can require a rollback segment as large as 1 GB.

Query Optimizer Settings

For the upgrade, use the query optimizer settings listed in 781927.1 (Article ID) My Oracle Support. This document was previously published as Siebel Technical Note 582 and Siebel Alert 1011.

Verifying Oracle Database Parameters for Multiple Processors in a Siebel Upgrade

Environments: Development, production test, production.

Databases: Oracle only.

If you have multiple processors (CPUs), then verify that the following parameters are set correctly. For information on settings, refer to Oracle Database documentation:

- `parallel_max_servers`. To enable use of multiple processors, this must be set to a number greater than 1.
- `parallel_min_servers`

Verifying the Oracle Database ODBC Definition for a Siebel Upgrade

Environments: Development, production test, production.

Databases: Oracle only.

When you install a Siebel Server, the installer creates the ODBC definition for the Oracle Database you specify.

When you upgrade your production test environment and your production environment, you must manually create an ODBC definition for connecting to your development environment database.

Verifying or Editing an ODBC Definition on Windows

Use the following procedure to verify or edit an ODBC definition on Windows.

To verify or edit an ODBC definition on Windows

1. Start the registry editor (regedit).

   > **CAUTION:** Editing the registry can adversely affect the operating system. Be sure you understand how to use regedit correctly.

2. Navigate to the following location: `HKEY_LOCAL_MACHINE\Software\Wow6432Node\ODBC\ODBC.INI\ODBC_Name`.

   ODBC_Name is the ODBC name for the Oracle Database.
3. In the data display pane, verify that the following entries are present:

Name	Type	Data
ColumnsAsChar	REG_SZ	1
ColumnSizeAsCharacter	REG_SZ	1

   If these entries are not present, then right-click in the data display pane, and choose New, and then String Value to add them.
4. Step off the ODBC_Name and return to it. Verify that the two new entries are present and correct.
5. Close the Registry Editor.

Verifying or Editing an ODBC Definition on UNIX

Use the following procedure to verify or edit an ODBC definition on UNIX.

To verify or edit an ODBC definition on UNIX

1. Navigate to the following file in the Siebel Server installation directory:

   ```
   $SIEBEL_ROOT/sys/.odbc.ini
   ```

2. Open the .odbc.ini file and add the following two entries:

   Set the ColumnsAsChar value to 1.

   Set the ColumnSizeAsCharacter to 1.
3. Save the file.
4. Stop and restart any processes that are using this .odbc.ini file.

8 Preparing a Microsoft SQL Server Database for a Siebel Upgrade

Preparing a Microsoft SQL Server Database fora Siebel Upgrade

This chapter provides initial preparatory information for a Siebel database upgrade on the Microsoft SQL Server database. This chapter includes the following topics:

- *Verifying Microsoft SQL Server Sort Order for a Siebel Upgrade*
- *Setting Microsoft SQL Server Temporary Space Size for a Siebel Upgrade*
- *Setting Microsoft SQL Server Configuration Parameters for a Siebel Upgrade*
- *Rebuilding Microsoft SQL Server Clustered Indexes for a Siebel Upgrade*

Verifying Microsoft SQL Server Sort Order for a Siebel Upgrade

Environments: Development, production test, production.

Databases: MS SQL Server only.

Platforms: Windows only.

Verify that the sort order of the master database and the database instance are the same. This prevents repository merge failure due to sort order mismatch.

Binary sort order is required for the development environment upgrade and strongly recommended for the production environment upgrades.

When you install Microsoft SQL Server, the collation method of the database instance is set by default to dictionary sort order. Every database you create thereafter inherits this setting.

When you create a database, you can accept the inherited sort order or specify the sort order. It is recommended that you set the sort order to binary at the Microsoft SQL Server instance level so that this sort order is inherited by newly created databases.

The sort order of the master database cannot be changed without rebuilding the instance. Consult your Microsoft documentation for instructions on setting database collation.

To verify that your database was created using a binary collation sequence

1. In the Query Analyzer window, enter the following command:

   ```
   sp_helpsort
   ```

   This command provides a sort order description.
2. Review the sort order description to verify binary sort order; for example:

   ```
   Latin1_General_BIN
   ```

   If you find that your Microsoft SQL Server database was not created using a binary collation sequence, then you must rebuild your database and reload your data. Review the Microsoft documentation for detailed instructions.

Setting Microsoft SQL Server Temporary Space Size for a Siebel Upgrade

Environments: Development, production test, production.

Databases: MS SQL Server only.

Platforms: Windows only.

Set the size of the database that Microsoft SQL Server uses for temporary space needed to execute queries.

To setup TEMPDB space

1. Make `TEMPDB` as big as the biggest table in the Siebel database, or half the size of the Siebel database.
2. Make sure that the files used by `TEMPDB` are configured to allow auto-growth.

   This allows Microsoft SQL Server to expand the temporary database as needed to accommodate upgrade activity. Alternatively, you can set MAXSIZE to the size of the biggest table or to 50% of the size of the Siebel database.
3. Consider putting `TEMPDB` on a separate drive to improve performance.
4. Execute `dbcc shrinkdatabase` against `TEMPDB`.

Setting Microsoft SQL Server Configuration Parameters for a Siebel Upgrade

Environments: Development, production test, production.

Databases: MS SQL Server only.

Platforms: Windows only.

This topic provides upgrade-specific settings for Microsoft SQL Server. Use the following strategy to set parameters:

- Set parameters using the recommendations in *Siebel Installation Guide* for the operating system you are using. Recommendations are located in the chapter on configuring the RDBMS.
- For the upgrade, revise the configuration parameters listed in the following table.
- After the upgrade, reset the configuration parameters to the values listed in *Siebel Installation Guide* for the operating system you are using.

The following table lists upgrade settings for Microsoft SQL Server database parameters. For parameters not listed in this table, it is recommended that you accept the default settings. Most of the parameter settings in the table are the default settings.

Parameter	Setting
Max. degree of parallelism	`1`
Cost threshold for parallelism	5
Fill factor (%)	`90`
Index create memory (KB)	`0`

For the Siebel database, set the following options to `ON` (enabled) for the upgrade:

- **truncate log on chkpt.** Set this option to `ON` (enabled) for upgrade only. Also, for upgrade only, execute the `alter` command against the Siebel database, specifying `set recovery simple`.
- **torn page detection.**
- **auto create statistics.**
- **auto update statistics.**
- **Database size.** Increase your database file size by resetting the `Autogrowth` parameter to between 25% and 50%. Failure to do this could diminish upgrade performance and possibly impact the success of your upgrade.
- For a full list of recommended settings for your postupgrade production environment, see the chapter on configuring the RDBMS in *Siebel Installation Guide* .

Rebuilding Microsoft SQL Server Clustered Indexes for a Siebel Upgrade

Environments: Development, production test, production.

Databases: MS SQL Server only..

Platforms: Windows only.

If you have large tables that you use extensively (such as `S_EVT_ACT`, `S_CONTACT, S_OPTY`, `S_OPTY_POSTN, S_ORG_EXT`), then use the Microsoft SQL Server `create index` command with `drop_existing` clause to rebuild large tables with high fill factor (60%-70%).

9 Preparing Siebel Application Data for Upgrade

Preparing Siebel Application Data for Upgrade

This chapter provides guidelines for preparing your Siebel Business Applications data for the Siebel database upgrade. This chapter includes the following topics:

Preparing Siebel Customized Seed Data for Upgrade

Environments: Development, production test, production.

Platforms: Windows, UNIX, IBM z/OS.

Modified seed data are not upgraded, even if the upgraded seed data are required for an application to run normally, with the following exceptions:

- If you have modified seed data through the user interface or through Siebel Enterprise Integration Manager, then the repository merge preserves the modifications using the same logic as for other repository objects.
- (Since Siebel CRM 8.0) Customizations to seeded workflows in the Prior Customer Repository are merged into workflows in the New Customer Repository by the repository merge process.

Before starting the upgrade, evaluate whether you must retain your seed data modifications. If practical, consider discarding the modifications. After the upgrade, you can reimplement those modifications that do not interfere with operation of applications.

Seed data are records provided in the Siebel database tables as part of a release. Seed data provides information needed to run a Siebel application. Examples of seed data are the values in a List of Values (LOV) definition, default mappings of views to responsibilities, and predefined queries.

Note: The paragraphs that follow provide a generalized description of how seed data for U.S. English (ENU) is imported during the upgrade process, including whether existing data is replaced. This description does not apply to all types of seed data, including seed data for non-ENU languages.

The ROW_ID value for ENU seed data records provided in a release begins with 0 (zero). In addition, these records have a default LAST_UPD value of 1980-01-01. The LAST_UPD column stores the date when the record was last updated.

If you modify an ENU seed data record, then the value of LAST_UPD is changed from 1980-01-01 to the date the modification was made. If you add a new seed data record, then the ROW_ID value does not begin with 0, and the LAST_UPD value is the record creation date.

When you perform the upgrep part of the upgrade, the dataimp utility upgrades ENU seed data records as follows:

1. For records where the ROW_ID value begins with 0, it erases ENU seed data records where the value for LAST_UPD is 1980-01-01, unless prevented by scripting.
2. Dataimp replaces these records with those contained in seed data files included in the release. In the new records, the value for LAST_UPD is 1980-01-01.
3. Dataimp does not erase and replace seed data records where the value for LAST_UPD is later than 1980-01-01. Instead, for each of these records, dataimp writes an error to the log file. The error is benign and does not cause the upgrade to fail. Use these error entries to verify which seed data records were not changed.

For ENU seed data records you have modified, you can discard the modifications by changing the value of LAST_UPD to 1980-01-01. This causes dataimp to replace these records with those from the new release. For seed data records you have created, the upgrade process retains these records.

About Deactivated LOVs

If you deactivated any standard Siebel LOVs in your prior release, then you must, before executing the UPDATE statements on the S_LST_OF_VAL table, make sure to reactivate the corresponding standard Siebel LOVs through the prior release's user interface. Standard Siebel LOVs that were manually deactivated through the Siebel user interface (by changing the Active value to N) will have a ROW_ID value beginning with 0, but the LAST_UPD value will no longer be 1980-01-01. For more information, see *Running the Siebel Database Configuration Wizard on Windows* or *Running the Siebel Database Configuration Wizard on UNIX*.

Discarding Seed Data Modifications

Use the following procedure to discard seed data modifications.

To discard seed data modifications

1. In Siebel Tools, select the Table object.
2. In the list applet, query for `"*` in the Seed Filter column.

   The query returns a list of tables containing seed data.
3. Use one of the following scripts to set LAST_UPD to 1980-01-01 for customized seed data records in these tables. In the scripts, tablename is the name of table containing seed data.

Database	Script
Oracle Database	`UPDATE tablename SET LAST_UPD = TO_DATE('1980-01-01', 'YYYY-MM-DD') WHERE ROW_ID LIKE '0%' AND LAST_UPD > TO_DATE('1980-01-01', 'YYYY-MM-DD')`
IBM DB2 and IBM DB2 for z/OS	`UPDATE tablename SET LAST_UPD = TIMESTAMP('1980-01-01-00.00.00') WHERE ROW_ID LIKE '0%' AND LAST_UPD > TIMESTAMP('1980-01-01-00.00.00')`

Database	Script
Microsoft SQL Server	`UPDATE tablename SET LAST_UPD = CONVERT(DATETIME,'1980-01-01') WHERE ROW_ID LIKE '0%' AND LAST_UPD > CONVERT(DATETIME,'1980-01-01')`

Related Topics

About the Siebel Database Upgrade Log Files

About the Siebel Repository Merge

Upgrade Planning for Siebel Workflow Designer

Migrating Siebel Household Data

Upgrades: Applies to Siebel Financial Services upgrades from 7.x that have retained the Siebel 6.x form of household associations.

Environments: Development, production test, production.

Platforms: Windows, UNIX, IBM z/OS.

Siebel CRM version 7.0.x introduced the Party model. This changed the way relationships between households and entities, such as activity and claim, are handled.

You have two options for migrating household data:

- Migrating household relationships to the Party model (recommended)
- Retaining the Siebel 6.x form of household relationships

Migrating Household Relationships to the Party Model

To check household data integrity and support migration of household data to the Party model, you must run the household verification script (`HH_MIG_populate.sql`).

The script verifies that at least the same number of entities will belong to a household after the upgrade as belong to it before the upgrade.

The household verification script makes the following assumptions:

- A household has at least one contact.
- The primary contact of a Policy/Financial Account is one of the contacts associated with this Policy/Financial Account.
- The primary contact of a Claim is one of the contacts associated with this Claim.
- The primary contact of an Opportunity is one of the contacts associated with this Opportunity.
- The primary contact of a Company is one of the contacts associated with this Company.

The script populates a temporary table with data, `TEMP_HH_OBJ_MIG` and generates a report based on an output file. Output is in the form of row IDs. The script verifies that every household associated with an entity includes a contact associated with that entity.

If there is no output, then this means data integrity is good, and no action is required. If you receive output, then you must examine the relationship between contacts and households.

To run the household verification script

1. Run one of the following household verification scripts as required:

   Windows:

   ```
   odbcsql /U Tableowner /P Password /S ODBCDataSource /a /c REM /separator / /
   O OutputFileLocation\HH_Mig_populate.txt
   /L LogFileLocation\HH_Mig_populate.log ScriptLocation\HH_Mig_populate.sql /v y
   ```

   UNIX:

   ```
   odbcsql /U Tableowner /P Password /S ODBCDataSource /a /c REM /separator / /O OutputFileLocation/
   HH_Mig_populate.txt
   /L LogFileLocation/HH_Mig_populate.log ScriptLocation/HH_Mig_populate.sql /v
   ```

   where:

   - `Tableowner` is the tableowner
   - `Password` is the tableowner password
   - `ODBCDataSource` is the data source of the database
   - `OutputFileLocation` is the location of the output file:
     - Windows: `SIEBEL_ROOT\log\HH_Mig_populate.txt`
     - UNIX:`$SIEBEL_ROOT/log/HH_Mig_populate.txt`
   - `LogFileLocation` is the location of the log file:
     - Windows: `SIEBEL_ROOT\log\HH_Mig_populate.log`
     - UNIX: `$SIEBEL_ROOT/log/HH_Mig_populate.log`
   - `ScriptLocation` is the location of the script:
     - Windows: `DBSRVR_ROOT\database_platform\HH_Mig_populate.sql`
     - UNIX: `DBSRVR_ROOT/database_platform/HH_Mig_populate.sql`

   Windows example:

   ```
   odbcsql /U Tableowner /P Password /S ODBCDataSource /a /c REM /separator / /O
   C:\sea7xx\siebsrvr\Log\HH_Mig_populate.txt /L
   C:\sea7xx\siebsrvr\Log\HH_Mig_populate.log
   C:\sea7xx\dbsrvr\DB2\HH_Mig_populate.sql /v y
   ```

2. If you receive output, then review the temporary table and check the following for each contact. Make corrections as needed:

   - Contact is correct and household is incorrect.
   - Contact is incorrect and household is correct.
   - Contact is incorrect and household is incorrect.

Preparing Siebel Mobile User Data for Upgrade

Environments: Development, production test, production.

Platforms: Windows, UNIX, IBM z/OS.

This topic applies primarily to developers running the Siebel Mobile Web Client in the development environment and to end users in the production environment. This topic applies to the production test environment only if it has Siebel Mobile Web Client users.

After synchronizing, mobile users must make no further changes to their local databases until the upgrade has been completed. Any changes made during the upgrade are lost when they are reinitialized following the upgrade.

Complete the following steps before beginning the upgrade of either a development environment or a production environment. For additional information on these steps, see *Siebel Remote and Replication Manager Administration Guide* and *Siebel System Administration Guide* .

To prepare Siebel Mobile Web client users for the database upgrade

1. Perform a partial synchronization for mobile users, sending all transactions to the server database.
2. Verify that Siebel Mobile Web Clients have synchronized and that all changes have been merged into the server database as follows:
   a. Check that no transaction files remain in the synchronization inbox and outbox for any mobile user. The synchronization inbox for each user is on the Siebel Server:
      Windows: `SIEBEL_ROOT\docking\MOBILEUSERNAME`
      UNIX: `$SIEBEL_ROOT/docking/MOBILEUSERNAME`
      Transaction files are in the format number.dx; for example, `00000023.dx`.
   b. Check the mobile users Remote Status view and resolve any insert conflicts.
   c. Log onto a Siebel Business Application, such as Call Center, as the Siebel administrator. Use the Tasks view from the Administration-Server Management screen to make sure that each Transaction Merger task has successfully completed.
   d. Verify that Workflow Monitor and Workflow Action agents have processed all pending requests. If Workflow Manager has completed successfully, then the `S_ESCL_REQ` table must not have any rows.
3. To prevent synchronization of Siebel Mobile Web Clients with the database server, stop or disable all Siebel Remote components on all Siebel Servers.
4. Disconnect all Siebel Web Clients from the Siebel Server by stopping the appropriate Application Object Managers, as described in *Siebel System Administration Guide* .

Preparing Siebel Address Data for Upgrade

Upgrades:

- From Siebel CRM 8.0.x (SEA) to Siebel CRM 8.1.1.x (SIA)
- From Siebel CRM 8.1.x (SEA) to Siebel CRM 8.1.1.x (SIA)

- From Siebel CRM 7.8.2 (SEA) to Siebel CRM 8.1.1 (SIA)

Environments: Production test, production.

Databases: All databases.

At Siebel CRM version 7.7, the way address data is stored changed. To prepare for the revised storage scheme, you must verify that there are no records with the same row IDs within or across the tables S_ADDR_PER and S_ADDR_ORG.

CAUTION: There must be no duplicate row IDs in these tables or the upgrade will fail.

To prepare address data for upgrade

1. Run `rpt_dup_addr_rowids.sql` against the Siebel database. The script is located in the following directory:
   Windows: `DBSRVR_ROOT\database_platform`
   UNIX: `DBSRVR_ROOT/database_platform`
   In these paths, database_platform is the database type, for example DB2.
2. Review the output generated by the script.
3. If the output contains records with duplicate row IDs, then use Siebel Enterprise Integration Manager or the application to delete unwanted records.
4. After addressing all the duplicate row IDs, rerun the script and verify there are no more duplicates.

Related Topic

Preparing Siebel Territory Management Rules for Upgrade

Preparing Siebel Territory Management Rules for Upgrade

Upgrades from: Siebel 7.8.x.

Environments: Production test, production.

Territory rules management was introduced in Siebel CRM 7.8. but since Siebel CRM 8.0, territory rules cannot have overlapping effective date ranges. For example, the following two rules have overlapping effective date ranges:

- **Rule 1**
  Territory: A100
  Account: XYZ Corp
  Effective Start Date: 1/1/2006
  Eff. End Date: 6/30/2006
- **Rule 2**
  Territory: A100

Account: XYZ Corp

Effective Start Date: 4/1/2006

Eff. End Date:

You must identify rules with overlapping effective date ranges and modify the rules to eliminate any overlap.

To prepare territory rules for upgrade

1. Run the following script against the database:

   Windows: `DBSRVR_ROOT\common\TM_DupRuleCheck.sql`

   UNIX: `DBSRVR_ROOT/common/TM_DupRuleCheck.sql`

   The script returns a list of all territory rules that have overlapping effective date ranges.
2. In Territory Management, revise rule effective date ranges to remove the overlaps.

Preparing Siebel Customizable Product Data for Upgrade

Environments: Production test, production.

Customizable Products in Work Spaces

The upgrade does not migrate unreleased customizable products in work spaces. If you want to migrate unreleased customizable products, then you must release them before the upgrade. This includes products with components and products with attributes.

Class Products

Verify that the Orderable flag is not set for class products. When this flag is not set, class products do not display as selectable products in quotes and orders after the upgrade.

The upgrade converts class products to a product and a product class. The upgrade sets the Product Class property for the product to Product Class.

10 Upgrading the Siebel Database

Upgrading the Siebel Database

This chapter provides guidelines for performing a Siebel database upgrade. It includes the following topics:

Modifying siebel.cfg Before Upgrading Siebel Database

Before upgrading to Siebel CRM 17.x or later, a number of parameters must be changed in the siebel.cfg file as shown in the following procedure, otherwise the Siebel database upgrade will fail when upgrading in upgphys mode.

The siebel.cfg file must also be updated correctly before running the Key Database Manager utility, which is used to maintain the key file (keyfile.bin). For more information on the parameters to set (or update) in siebel.cfg before running the Key Database Manager utility, see *Siebel Security Guide* .

To modify siebel.cfg

1. Open siebel.cfg, which is located in the `$SIEBEL_HOME\siebsrvr\bin\` folder and modify the following parameters:

   - Change: `ClientRootDir = siebel-root\ses`

     To the following: `ClientRootDir = siebel-root\ses\siebsrvr.`
   - Change: `ServerDbODBCDataSource = %MASTER_DATASOURCE%`

     To the following: `ServerDbODBCDataSource = odbc connection name.`
   - Change: `DataSourceName = $(DefaultDataSource)`

     To the following: `DataSourceName = ServerDataSrc.`
2. (Optional) Resume the upgphys process from the command line as follows:

   > **Note:** This step applies only if you have already run upgphys and received an encryption error similar to the following: *INFO:Recommendation : Verify cfg file(C:\Siebel\15.0.0.0.0\ses\siebsrvr\bin\enu\siebel.cfg).*

   a. Go to the `$SIEBEL_HOME\siebsrvr\bin\enu` folder.
   b. Enter one of the following commands as required:

      - Windows: `siebupg /m master_UPGRADEOPTION_ENVIRONMENT_VERSION.ucf`
      - UNIX: `srvrupgwiz /m master_UPGRADEOPTION_ENVIRONMENT_VERSION.ucf`

   The upgphys process will continue to run

Creating a New ODBC Data Source Before Upgrading Siebel Database

When you execute a migration installation for Siebel Enterprise Server (SES), the existing ODBC data source is migrated from the previous to the new Siebel CRM version. When an OS and/or hardware replacement is warranted before Siebel CRM installation, you must install the latest monthly updates into the newly installed environment. This scenario requires that a new ODBC data source be created manually, prior to running the database upgrade, since an ODBC data source has not been generated yet.

To create a new ODBC data source before upgrading Siebel Database

1. On Windows, complete the following tasks:

   - *Preparing Siebel Tools for Custom ODBC Data Source Names on Oracle Database*
   - *Preparing Siebel Tools for Custom ODBC Data Source Names for All Databases*
2. On UNIX/Linux, copy the templ_oracle.odbc.ini file located under the `siebsrvr\sys` directory and make the necessary changes.

Example of a Siebel Development Environment Upgrade Flow

This topic presents the flow of steps in part of a typical development environment upgrade. The steps are extracted from an actual driver file. To perform an upgrade, the Upgrade Wizard reads the steps in a driver file and performs the commands the steps contain. The driver file type used in this example is as follows:

- Upgrade: Siebel Industry Application (SIA) 8.0 to Siebel SIA 8.1
- Environment: Development
- Upgrade mode: upgrep
- Database: Oracle Database
- Multilingual: No

The following table lists the steps in the driver file. The Script or Input File column in the following table lists the SQL file or input file that is executed in each step. The Comment column provides a brief explanation of what the SQL file or input file does. The SQL files used for an upgrade and the contents of the SQL files vary depending on the upgrade mode and database.

Step	Script or Input File	Description
Determine collation sequence of database.	Not applicable	Determines database sort order.
Verify repository name	rename_existing_repositories.sql	Renames Siebel Repository to Prior Customer Repository.
Remove interface tables	dropif-db.sql	Removes all Siebel Enterprise Integration Manager tables.
Remove database triggers	trigdrop-db.sql	Removes all dynamically created triggers.
Remove database-level functions and procedures	drop_db_func_proc.sql	Removes the exchange rate function: exrate.
Prepare for table creation	pret.sql	Removes specified tables. Performs DDL operations such as adding columns to tables. Performs DML operations such as revising date formats.
Create temporary tables for SIA	ddlimp utility ddl_temp_sia.ctl as input	The input file specifies the structure of the tables to be created or updated. These tables are used to perform data migration and other DML changes.
Prepare for table creation for SIA	pret_sia.sql	Drops specified tables. Performs DDL operations such as adding columns to tables.

Step	Script or Input File	Description
		Performs DML operations such as revising date formats.
Create and update tables	ddlimp utility ddl.ctl as input	The ddl.ctl file specifies the structure of tables to be created or updated.
Create temporary tables for stored procedures	ddlimp utility ddlsptbl.ctl as input	The input file specifies the structure of temporary tables to be created or updated.
Household data migration for FINS	Household_Mig_Fins.sql	Creates and populates specified temporary tables. Then migrates data to them and performs DML operations. Migrates data back to primary tables. Drops temporary tables.
Prepare for index creation	preschm.sql	Performs DML operations. Moves data between tables. Changes data in existing fields based on specified conditions.
Prepare for index creation for SEA	preschm_sea.sql	Same as preschm.sql.
Prepare for index creation for SIA	preschm_sia.sql	Same as preschm.sql.
Create indexes	ddlimp utility ddl.ctl as input	The input file specifies the structure of indexes to be created.
Delete old license key	delappkey.sql	Deletes the Siebel license key from S_APP_ KEY.
Preparation of prior customer repository	SWTClob.jar	The SWT to OD Conversion Utility converts Siebel Web Templates to an Object Definition Layout. Converted Web templates are stored in the database.
Import seed data	dataimp utility seedupg0.inp as input seedupg1.inp as input seedupg_locale.inp as input	Prior to importing seed data, dataimp deletes existing seed data. The seedupg* files contain filters that dataimp uses to prevent deleting seed data that you have modified or seed data meeting specified criteria. Unmodified seed data has a last update date (LAST_UPD) of 1980-01-01. Dataimp does not delete records where LAST_UPD is later than this date.
Upgrade data after seed data import	upg_data_afterseed.sql	For customers who have not converted to UTC time, sets the UTC value in S_SYS_PREF to

Step	Script or Input File	Description
		False. For customers who have converted to UTC time, the script takes no action.
Upgrade data after seed data import SIA	upg_data_afterseed_sia.sql	None.
Set system preference for codepage for DB	set_codepage.sql	Sets the database codepage in the S_SYS_PREF.
Set system preference for Unicode codepage for DB	set_unicode.sql	Sets the Unicode codepage to UTF-8 in S_SYS_PREF.
Update version component information	upd_upgcomp.sql	Updates the S_UPG_COMP table with the product release level. The S_UPG_COMP table stores version information for application executable programs.
Run Oracle-specific DDL commands	ddlora.sql	Creates Oracle-specific DDL information, such as default storage parameters for docking objects, repository objects, and seed objects.
Import common ancestor repository	repimexp utility Standard Siebel Repository as input	Imports the Standard Siebel Repository into S_REPOSITORY. For example, if you are upgrading from Siebel 8.1.x, then this command imports the standard Siebel 8.1.x repository.
Remove EIM columns and indexes	rmv_anc_eim_proc_col_ind.sql	Removes Siebel Enterprise Integration Manager processing columns and indexes from the Prior Customer Repository and the common ancestor repository (Standard Siebel Repository). This prevents the repository merge from preserving Siebel Enterprise Integration Manager columns incorrectly. The merge will preserve only those Siebel Enterprise Integration Manager columns shipped with the new release.
Update Siebel database version	update_ver.sql seeduver.sql	The update_ver.sql script creates a temporary table, S_APP_VER_TEMP, which contains new version information for the database schema. The seeduver.sql script updates S_APP_VER with this information.
Import New Customer Repository	repimexp utility Imports New Customer Repository	Imports the New Customer Repository into S_REPOSITORY. Revises schema version information in S_APP_VER.
Encryption Upgrade	EncryptionUpgrade.jar	Siebel Business Applications allow customers to encrypt sensitive information stored in

Step	Script or Input File	Description
		the Siebel database, for example, credit card numbers, Social Security numbers, birth dates. This information cannot be viewed without access to Siebel Business Applications. Sensitive data can be encrypted by using AES (Advanced Encryption Standard). This utility identifies the RC2 encrypted columns and upgrades their data to the AES. This utility updates the logical layer of data for columns which are candidates for encryption.
Restore database version	restore_ver.sql	Uses S_APP_VER_TEMP to update the schema version information in S_APP_VER. Drops S_APP_VER_TEMP.
Upgrade repository data SIA	repos_upgrade_sia.sql	None.
Upgrade repository data	repos_upgrade.sql	Makes specific repository-related changes to repository records and to other tables.
Set repository workflow domains to MLOV	set_multilingual.sql	None.
Install SQL packages	seeduver.sql	Verifies that versions are set correctly in S_ APP_VER.
Install SQL packages	ifstrg.sql	Sets storage parameters for Siebel Enterprise Integration Manager tables.
Install SQL packages	ifindxstrg.sql	Sets storage parameters for Siebel Enterprise Integration Manager table indexes.
Install SQL packages	pkgseq.sql	Adds a suffix to row IDs in the S_SEQUENCE table. Ensures that row IDs are unique.
Install SQL packages	pkgldel.sql	Defines s_txn_log_del_proc. Procedure periodically deletes transactions from S_ DOCK_TXN_LOG. Also deletes rows from S_DOCK_TXN SET. Prevents need for large rollback segment.
Install SQL packages	trgreset.sql	Ensures that denormalized rows in S_TERR have correct values.
Install SQL packages	ddlseq.sql	Sets sequence numbers for specified tables.
Install SQL packages	pkgvis.sql	Creates function that modifies how Oracle optimizer does visibility check.

Step	Script or Input File	Description
Install SQL packages	delete_dock_rules.sql	Deletes Prior Customer Repository routing rules from S_DOC_VIS_RULE that meet specified conditions. Attempts to preserve rules added using Docking Wizard.
Create database-level functions and procedures	db_func_proc.sql	Creates or replaces the currency exchange rate function: exrate.
Set primary children in data tables	gen_primary1.sql	Sets primary child for S_DOC_QUOTE.
Set primary children in data tables	gen_primary2.sql	None.
Set primary children in data tables	gen_primary3.sql	Sets primary child for S_LOY_PROMO.
Set primary children in data tables	gen_primary4.sql	None.
Fix column alignment for custom objects	AlignApplet.jar	Applet alignments are executed based on the data type of the field. The following alignments are executed across the entire Repository for a similar look and feel for fields: • Consistent Left • Right • Center

Related Topics

About the Siebel Database Configuration Wizard Utilities

About the Siebel Upgrade Wizard and Driver Files

Renaming the Siebel Repository

Environments: Development environment only. (Also Production environment on UNIX.)

Platforms: Windows, UNIX, IBM z/OS.

Note: To prevent a naming conflict, before you run the upgrade, you *must* rename your existing development repository (*Siebel Repository*) to *Prior Customer Repository*. After the upgrade, your new development repository is given the name *Siebel Repository*.

To rename the repository

1. Start Siebel Tools and connect to the Siebel database.

   Use the version of Siebel Tools for the Siebel CRM release from which you are upgrading.
2. If you archived repository objects as .sif files, and you want to have them available in your application, then import these archive files back into the repository.

   If you do not check these objects back into the repository, then they will not be upgraded. You need only to check in those archived objects that you need in the future and want to have available in your upgraded application.
3. From the View menu, choose Options.
4. Click the Object Explorer tab.

   The Object Explorer hierarchy displays.
5. Locate Repository in the list, put a check mark in the adjacent box, and then click OK.

   This exposes the repositories.
6. In the Object Explorer, click the Types tab, and then Click Repository.
7. In the Repositories list view, verify that your existing repositories do not use the names reserved for the upgrade process:

   - **New Customer Repository**
   - **New Siebel Repository**
8. Locate your current Siebel Repository in the list applet.
9. Click the name and change it to `Prior Customer Repository`.

   For more information about renaming repositories, see *Configuring Siebel Business Applications* .
10. Step off the list to commit the record to the database.

    If the validation check fails, then verify that you have renamed the repository correctly.

Ancestor Repositories

This task is a step in *Process of Meeting Requirements for an Incremental Repository Merge*.

For any of the upgrade paths, described in *Supported Upgrade Paths for Siebel CRM*, that include a database upgrade or an incremental repository merge, a copy of the Siebel Repository that was released in conjunction with the old release is required to perform the upgrade or merge.

The ancestor repositories are distributed with the Siebel installer, but are only installed in the event that the user selects them during the installation process.

If this step was not performed during the Siebel installation process, then re-run the installer and add the Ancestor Repositories before attempting to continue with the upgrade.

Changing the Siebel Database Configuration Utilities Language

Environments: Development, production test, production.

Platforms: Windows, UNIX, IBM z/OS.

The Database Configuration Utilities launch in the language selected when you ran the Siebel Enterprise Server Installer. You can change the language in which the utilities run, if desired, from the language chosen during installation.

To change the Database Configuration Utilities language, see the *Siebel Installation Guide* .

If you want an additional language to appear in the language list in the Database Configuration Utilities, then you first must install the appropriate language pack on the database server and on the Siebel Server. For information about installing additional language packs, see the *Siebel Installation Guide* .

Related Topic

About the Siebel Database Configuration Wizard Utilities

Preparing to Run the Siebel Database Configuration Wizard

Environments: Development, production test, production.

Use this topic to identify the information you must enter when running the Database Configuration Utilities. Collect this information, and verify that it is correct before running the utilities.

> **Note:** If you are running Siebel CRM version 8.1.1 (SEA), and want to migrate to Siebel CRM version 8.1.1 (SIA), then you do so with the Database Configuration Wizard.

The Database Configuration Utilities are a group of wizards that request information about the upgrade process you want to perform. The utilities add this information to a primary upgrade configuration file and call an SQL generator. The SQL generator uses the information to create or populate SQL files:

- The following table lists the information that the utilities request for performing an upgrade.
- The table on the following page lists the additional information specific to Additive Schema Changes mode.

After the Database Configuration Wizard exits you run the Upgrade Wizard. The Upgrade Wizard executes the SQL files against the Siebel database.

Field Name or Menu	Required Information
Siebel Server Directory	The absolute path of the directory where the Siebel Server is installed. For UNIX, do not enter the string `$SIEBEL_ROOT`.

Field Name or Menu	Required Information
Siebel Database Server Directory	The absolute path of the directory where the Siebel Database Server (Siebel Database Configuration Utilities) is installed. For example: `C:\sba81\dbsrvr`.
RDBMS Platform	Choose the RDBMS type: IBM DB2, Microsoft SQL Server, or Oracle Database. **Note:** For IBM DB2 for z/OS, see *Siebel Database Upgrade Guide for DB2 for z/OS* instead of this guide.
Siebel Database Operation menu	For upgrep, upgphys and Prepare for Production modes, choose Upgrade Database. For Apply Additive Schema Changes mode, choose Apply Additive Schema Changes. The remaining menu choices are for database installation and administration.
Environment Type	Choose Development for development environment upgrades. Choose Production for production test environment and production environment upgrades.
Upgrade Options	Choose one of the following: • Development Environment: Upgrade Siebel Database Schema (upgrep) Upgrade Custom Database Schema (upgphys) • Production Environment: Upgrade Siebel Database Schema (upgrep and upgphys)
Siebel Industry Application	As appropriate for the environment you are upgrading from, choose SIA (for Siebel Industry Applications) or SEA (for Siebel Business Applications, also known as Siebel Cross-Industry Applications).
Current Siebel Version	Choose the option for the application version that you are upgrading your Siebel database from. For more information about valid options for performing a full database upgrade, see *Supported Upgrade Paths for Siebel CRM*. For information about specifying options for performing an Incremental Repository Merge instead, see *Performing a Siebel Incremental Repository Merge*.
Database Encoding	Indicate whether your database uses a Unicode code page.
ODBC Data Source Name	Verify the ODBC name for connecting to the Siebel database that you are upgrading. If the ODBC name is not correct, then enter the correct name.
Database User Name and Database Password	Account name and password for the Siebel administrator of the Siebel database that you are upgrading. **Note:** For more information about supported characters for Siebel passwords, see *Siebel Security Guide* .

Field Name or Menu	Required Information
Database Table Owner and Database Table Owner Password	Account name and password for the Siebel database table owner. **Note:** For more information about supported characters for Siebel passwords, see *Siebel Security Guide* .
Index Table Space Name and Table Space Name	Oracle Database and IBM DB2 only. Index tablespace name and tablespace name (4-KB tablespace name for IBM DB2).
16-KB Table Space Name, 32K Table Space Name	IBM DB2 only. The 16-KB and 32-KB tablespace names.
Database Server OS	Choose the RDBMS server operating system type.
Parallel Indexing	Oracle Database only. Select parallel indexing if you want SQL commands for index creation to include the arguments parallel and no logging. Parallel indexing causes an index to be created using parallel processing, which requires an RDBMS server with multiple processors. Verify with your database administrator whether your RDBMS server is configured for parallel processing. **Tip:** Oracle Library search phrase: parallel execution. Selecting parallel indexing does not cause multiple indexes to be created simultaneously, in parallel. To set up parallel indexing, you must set up parallel index-creation threads using Siebel Upgrade Tuner. You create parallel threads as part of tuning the production upgrade files. For more information, see *Tuning the Siebel Upgrade Files*
Security Group ID/Grantee	Security group or grantee name for Siebel application users. Must have select, update, insert, and delete privileges on Siebel application tables. Specify SSE_ROLE.

Custom Scripts Directory	During the upgrade process, the Repository Sanitization script is executed. This script scans the prior customer repository to check if there are any unreferenced repository objects across multiple applications. The unused objects are then inactivated. If a repository object is referred to in a custom script, it will not be a candidate for inactivation.
	
	Custom files have to be copied from the following locations:
	
	• For Siebel CRM 15.0 and prior releases:
	
	`<SWSE HOME>\public\<LANG>\<BUILD#>\scripts\siebel\custom`
	
	• For Siebel CRM 16.x:
	
	`<SWSE HOME>\public\scripts\<LANG>/siebel/custom`
	
	In addition, copy every language required into a directory and specify that location.

Web Templates Directory	Directory where the custom web templates are stored. From Siebel CRM 17.0 onwards, web templates are stored in the database. Since custom web templates must be moved to the database as well, a step is executed during the upgrade process that moves all the web templates specified in this particular location to the database.

Field Name or Menu	Required Information
	You must copy over the custom files from previous releases from: **`<SES>\siebsrvr\webtempl`** Copy all the custom files into a single directory and then specify that location in the upgrade path when prompted.
Verify Repository After Upgrade	Indicate whether you want to execute the steps to verify the repository during upgphys. To perform upgphys separately, select the Verify Repository After Upgrade option in the Database Configuration Wizard.
Upgrep log directory	If you select the option Verify Repository After Upgrade in the previous step, then you will have to provide the log directory of the upgrep process. The log directory is of the form: **`$SIEBEL_ROOT/siebsrvr/log/upgrep_dev_UpgradeNumber`** For example: **`C:\ses\siebsrvr\log\upgrep_dev_811`** The **`log`** directory path is a requirement for generating the seed data conflict report.
Log Output Directory	Specify a different subdirectory, under the **`log`** directory, in which to create the log files. Accept the default or enter the directory name. If the directory does not exist, then it will be created. Do not use special characters, such as spaces or slashes.
Select runupg option	Indicate whether you want to run the operation you configured or run it at another time.

Additional Information Required for Apply Additive Schema Changes Mode

You can optionally run the Database Configuration Utilities in Apply Additive Schema Changes mode. Additive Schema Changes generates an SQL script, **`schema.additive.sql`**. The script contains production upgrep schema changes that can be performed while the database is online. This reduces the amount of time required for the upgrep when the database is offline.

The additional information shown in the following table is required when you run the Database Configuration Utilities in Apply Additive Schema Changes mode.

Field Name	Required Information
Schema File	The absolute path to the **`schema.ddl`** file. If a path is displayed, then verify that **`schema.ddl`** is located in this directory.
Output Directory	The absolute path to the directory where **`schema.additive.sql`** will be placed after the Upgrade Wizard generates it. The Upgrade Wizard also places the log file in this directory.

Related Topics

About the Siebel Database Configuration Wizard Utilities

About Siebel Additive Schema Changes Mode

Running the Siebel Database Configuration Wizard on Windows

Environments: Development, production test, production.

Platforms: Windows only.

For a description of information required to run the Database Configuration Wizard, see the following table.

Run the Database Configuration Wizard to upgrade the Siebel database. The Wizard collects information, populates a primary upgrade configuration file, and calls the SQL generator to create SQL commands. The Upgrade Wizard then uses the configuration file and SQL commands to upgrade the Siebel database.

Requirements

- Collect the information that the Database Configuration Wizard requires. See *Preparing to Run the Siebel Database Configuration Wizard*.
- Install the new release's languages packs for all deployed languages. See *Upgrade Planning for Multilingual Siebel Deployments*.
- Run the Database Configuration Wizard as described in this topic. All customers must perform this procedure. This procedure allows you to determine whether you have deployed languages that are not shipped with the Siebel product, or have unintended languages within your system that must be removed. For a list of shipped languages, see 1513102.1 (Article ID) on My Oracle Support.

  Note: If this procedure fails due to the presence of unshipped languages within your system, then you will receive an error message stating that your present installation was found to be incomplete. You are also displayed a list of languages which caused the error.

  If your system only contains shipped languages, and you still receive this error message, then you must review the records in the S_LST_OF_VAL database table using Siebel Tools. For more information, see *Preparing for a Multilingual Upgrade*.
- If applicable, perform the tasks in *Process of Deleting Unshipped Languages from the S_LST_OF_VAL Table on Windows*.

  CAUTION: Perform the tasks in this process only if your configuration operation failed due to unshipped languages being deployed within your system.

  This process allows you to run the configuration utility, validate all data in shipped languages, and pass over all data in unshipped languages. You can manually verify the data in unshipped languages following the successful completion of the Database Configuration Wizard.

For more information about starting the Database Configuration Wizard, see the *Siebel Installation Guide for Microsoft Windows* .

To run the Database Configuration Wizard on Windows

1. Ensure that no server tasks including the Siebel Gateway Name Service are running in the background.

   To verify, navigate to Start, Settings, Control Panel, and then Services.
2. Start the Database Configuration Wizard by selecting Start, Programs, Siebel SES Configuration, and then Configure DB Server.

   The first window of the Database Configuration Wizard appears.
3. Enter the information requested in each screen and click Next.
4. After you have entered all the requested information, the wizard displays a screen that lists the values you entered. If you must make changes, then click Back.
5. When the window displays inquiring whether you want to start the Upgrade Wizard, do the following:
   - **Development Upgrep mode:** Answer No. Do not start the Upgrade Wizard until you have performed all requirements.
   - **Development Upgphys mode:** Answer Yes to start the Upgrade Wizard.
   - **Combined Production Upgrep and Upgphys mode:** Answer No. Do not start the Upgrade Wizard until you have performed all requirements.
   - **Additive Schema Changes:** You will not be prompted. The Upgrade Wizard starts automatically and creates the `schema.additive.sql` script.
   - **Prepare for Production mode:** Answer Yes to start the Upgrade Wizard.

Just before displaying the prompt, the wizard calls the SQL generator to create or populate SQL scripts.

Process of Deleting Unshipped Languages from the S_LST_OF_VAL Table on Windows

To delete unshipped languages from the S_LST_OF_VAL table on Windows, perform the following tasks.

CAUTION: Perform the tasks in this process only if your configuration operation failed due to unshipped languages being deployed within your system.

- *Backing Up the S_LST_OF_VAL Table on Windows*
- *Deactivating Records for Unshipped or Unwanted Languages on Windows*
- *Importing Newly Created Records from the Data File on Windows*

Backing Up the S_LST_OF_VAL Table on Windows

This task is a step in *Process of Deleting Unshipped Languages from the S_LST_OF_VAL Table on Windows*. You back up your original S_LST_OF_VAL table before you begin deleting obsolete or unshipped languages. You can later import languages from this data file.

To back up the S_LST_OF_VAL table on Windows

1. In Siebel Tools, back up the original S_LST_OF_VAL table.

2. From the Windows command prompt, navigate to the `siebsrvr\bin` directory.
3. From this directory, execute a command like the following to run the Siebel dataexp utility (replacing $ tokens with actual values that represent the environment):

   ```
   dataexp /u $UserName /p $Password /c "$ODBCDataSource" /d $SiebelTableOwner /f
   $DataFileToExport.dat /l $LogFile.log
   ```

   The utility connects to the database, and displays the following prompt:

   ```
   Connecting to the database...
   Connected.
   Exporting Tables
   Enter table name:
   ```

4. Type S_LST_OF_VAL, and hit Enter.

   When the action is completed, the utility will display the following message:

   ```
   Exporting Tables
   Enter table name: S_LST_OF_VAL
   Reading tables matching "S_LST_OF_VAL" ...
   Exporting table S_LST_OF_VAL ... exported 30102 rows
   Enter table name:
   ```

5. Close the command window and check the log file for errors, to verify that the data file was created.

Deactivating Records for Unshipped or Unwanted Languages on Windows

This task is a step in *Process of Deleting Unshipped Languages from the S_LST_OF_VAL Table on Windows*. You render unshipped or obsolete languages inactive.

To deactivate records for unshipped or unwanted languages on Windows

1. Connect to the database using a database client that allows you to run SQL interactively (either one of the native clients, or Siebel odbcsql), and execute the following query:

   ```
   update S_LST_OF_VAL set ACTIVE_FLG = 'N'
   where LANG_ID not in ('ENU', 'lang_code1', ..., 'lang_codeN');
   ```

   Note: Be aware that lang_code* tokens must be replaced with actual codes of the languages that are going to be upgraded (such as FRA, DEU, and so on).

   CAUTION: You must not deactivate ENU records, even if you do not have the ENU language deployed in your environment. Some system records in S_LST_OF_VAL are ENU.

2. Choose commit (if your database client is not in auto-commit mode), and execute the following query:

   ```
   delete from S_LST_OF_VAL where ACTIVE_FLG = 'N';
   ```

3. Choose commit and exit the database client.

Importing Newly Created Records from the Data File on Windows

This task is a step in *Process of Deleting Unshipped Languages from the S_LST_OF_VAL Table on Windows*. Import the data file that you created when you backed up the S_LIST_OF_VAL table. This procedure is required to upgrade additional languages (those that were not available or were skipped during the primary upgrade).

Note the following:

- Depending on which data you have exported, how you set up the input file, and which options you use on the command line, you can import data for multiple languages at once or for one language at a time, or import data based on other criteria.
- In order to import data for only particular languages, for example, you must first use filtering mechanisms when you export the data, so that only those languages are to be included in the data file.
- If you use the /e N command-line option for dataimp, then dataimp only imports the data that was originally exported from the tables specified in the input file. If you instead use /e Y, then the dataimp imports the contents of the entire data.

To import newly created records from the data file on Windows

1. Create a text file that will serve as the input file for the Siebel dataimp utility.

   The contents of the file must be formatted as follows (replacing lang_code tokens with actual values that represent the environment):

   ```
   S_LST_OF_VAL
    where LANG_ID in ('lang_code1', ..., 'lang_codeN')
   ```
2. Save the file with the extension .INP.
3. From the Windows command prompt window, navigate to the `siebsrvr\bin` directory.
4. From the `siebsrvr\bin` directory, execute a command like the following to run the Siebel dataimp utility (replacing $ tokens with actual values that represent the environment):

   ```
   dataimp /u $UserName /p $Password /c "$ODBCDataSource" /d $SiebelTableOwner /f
   $DataFileName.dat /h Log /e N /i $InputFileName.inp /A 1
   ```
5. Check the log file for errors.
6. Connect to the database using a database client that allows you to run SQL interactively, to verify that the imported records are there.

   For example, you execute the following query:

   ```
   select count (LANG_ID), ACTIVE_FLG from S_LST_OF_VAL where LANG_ID = 'lang_code'
   group by ACTIVE_FLG;
   ```

   If the result set returns both active and inactive records, then you must investigate and see why some records have been deactivated, and either re-activate or delete these inactive records.

Related Topics

About the Siebel Database Configuration Wizard Utilities

Starting the Siebel Upgrade Wizard

Running the Siebel Database Configuration Wizard on UNIX

Environments: Development, production test, production.

Platforms: UNIX only.

Run the Database Configuration Wizard to upgrade the Siebel database. The Wizard collects information, populates a primary upgrade configuration file, and calls the SQL generator to create SQL commands. You then run the Upgrade Wizard to upgrade the Siebel database.

Prior to running the Database Configuration Wizard you must install the new release's languages packs for each language that you deploy.

Requirements

- Collect the information that the Database Configuration Wizard requires. See *Preparing to Run the Siebel Database Configuration Wizard*.
- Install the new release's languages packs for all deployed languages. See *Upgrade Planning for Multilingual Siebel Deployments*.
- Run the Database Configuration Wizard as described in this topic. All customers must perform this procedure. This procedure allows you to determine whether you have deployed languages that are not shipped with the Siebel product, or have unintended languages within your system that must be removed. For a list of shipped languages, see 1513102.1 (Article ID) on My Oracle Support.

  > **Note:** If this procedure fails due to the presence of unshipped languages within your system, then you will receive an error message stating that your present installation was found to be incomplete. You are also displayed a list of languages which caused the error.

  If your system only contains shipped languages, and you still receive this error message, then you must review the records in the S_LST_OF_VAL database table using Siebel Tools. For more information, see *Preparing for a Multilingual Upgrade*.
- If applicable, perform the tasks in *Process of Deleting Unshipped Languages from the S_LST_OF_VAL Table on UNIX*.

  > **CAUTION:** Perform the tasks in this process only if your configuration operation failed due to unshipped languages being deployed within your system.

  This process allows you to run the configuration utility, validate all data in shipped languages, and pass over all data in unshipped languages. You can manually verify the data in unshipped languages following the successful completion of the Database Configuration Wizard.

For more information about starting the Database Configuration Wizard, see the *Siebel Installation Guide for UNIX* .

To run the Database Configuration Wizard on UNIX

1. Verify that all servers are stopped:
    - Stop all Siebel Servers.
    - Stop the Siebel Gateway.
2. Make `$SIEBEL_ROOT` the current directory.
3. Source the environment variables from the `siebsrvr` root directory: `install_location/siebsrvr`:

    Korn shell: `. siebenv.sh`

    C shell: `source siebenv.csh`
4. Review the values of the following environment variables and confirm the settings are correct:
    - `SIEBEL_ROOT`. This path must end in `siebsrvr`. For example, `/usr/siebel/siebsrvr`.
    - `LANGUAGE`. This is the language in which the Database Configuration Wizard runs. The value of this variable is a language identifier string. For example, `enu` is the identifier string for English.

      If either `$SIEBEL_ROOT` or `$LANGUAGE` is not set or is incorrect, then you must correct them before proceeding.
5. Start the Database Configuration Wizard:

    ```
    install_location/config/config -mode dbsrvr
    ```
6. Enter the information requested in each screen and click Next.

    After you have entered all the requested information, the wizard displays a screen that lists the values you entered.
7. If you must make changes, then click Back.
8. If you are performing a development upgrep or combined production upgrep and upgphys, then do not start the Upgrade Wizard. Instead, return to the upgrep process checklist and perform the next steps.
9. If you are performing any of the following, then enter the command to start the Upgrade Wizard:
    - **Development environment upgphys.**
    - **Additive Schema Changes.** The Upgrade Wizard creates the `schema.additive.sql` script.
    - **Prepare for Production.**
10. The Database Configuration Wizard will now exit and prompt you to launch the Upgrade Wizard (srvrupgwiz).

Process of Deleting Unshipped Languages from the S_LST_OF_VAL Table on UNIX

To delete unshipped languages from the S_LST_OF_VAL table on UNIX, perform the following tasks.

CAUTION: Use the tasks in this process only if your configuration operation failed due to unshipped languages being deployed within your system.

- *Backing Up the S_LST_OF_VAL Table on UNIX*
- *Deactivating Records for Unshipped or Unwanted Languages on UNIX*
- *Importing Newly Created Records from the Data File on UNIX*

Backing Up the S_LST_OF_VAL Table on UNIX

This task is a step in *Process of Deleting Unshipped Languages from the S_LST_OF_VAL Table on UNIX*. You back up your original S_LST_OF_VAL table before you begin deleting obsolete or unshipped languages. You can later import languages from this data file.

To back up the S_LST_OF_VAL table on UNIX

1. In Siebel Tools, back up the original S_LST_OF_VAL table.
2. On the Siebel Server computer, navigate to the `SIEBSRVR_ROOT\bin` directory.
3. From this directory, execute a command like the following to run the Siebel dataexp utility (replacing $ tokens with actual values that represent the environment):

   ```
   dataexp /u $UserName /p $Password /c "$ODBCDataSource" /d $SiebelTableOwner /f
   $DataFileToExport.dat /l $LogFile.log
   ```

   The utility connects to the database, and displays the following prompt:

   ```
   Connecting to the database...
   Connected.
   Exporting Tables
   Enter table name:
   ```

4. Type S_LST_OF_VAL, and hit Enter.
   When the action is completed, the utility will display the following message:

   ```
   Exporting Tables
   Enter table name: S_LST_OF_VAL
   Reading tables matching "S_LST_OF_VAL" ...
   Exporting table S_LST_OF_VAL ... exported 30102 rows
   Enter table name:
   ```

5. Check the log file for errors, to verify that the data file was created.

Deactivating Records for Unshipped or Unwanted Languages on UNIX

This task is a step in *Process of Deleting Unshipped Languages from the S_LST_OF_VAL Table on UNIX*. You render unshipped or obsolete languages inactive.

To deactivate records for unshipped or unwanted languages on UNIX

1. Connect to the database using a database client that allows you to run SQL interactively (either one of the native clients, or Siebel odbcsql), and execute the following query:

   ```
   update S_LST_OF_VAL set ACTIVE_FLG = 'N'
   where LANG_ID not in ('ENU', 'lang_code1', ..., 'lang_codeN');
   ```

> **Note:** Be aware that lang_code* tokens must be replaced with actual codes of the languages that are going to be upgraded (such as FRA, DEU, and so on).

> **CAUTION:** You must not deactivate ENU records, even if you do not have the ENU language deployed in your environment. Some system records in S_LST_OF_VAL are ENU.

2. Choose commit (if your database client is not in auto-commit mode), and execute the following query:

```
delete from S_LST_OF_VAL where ACTIVE_FLG = 'N';
```

3. Choose commit and exit the database client.

Importing Newly Created Records from the Data File on UNIX

This task is a step in *Process of Deleting Unshipped Languages from the S_LST_OF_VAL Table on UNIX*. Import the data file that you created when you backed up the S_LIST_OF_VAL table. This procedure is required to upgrade additional languages (those that were not available or were skipped during the primary upgrade).

Note the following:

- Depending on which data you have exported, how you set up the input file, and which options you use on the command line, you can import data for multiple languages at once or for one language at a time, or import data based on other criteria.
- In order to import data for only particular languages, for example, you must first use filtering mechanisms when you export the data, so that only those languages are to be included in the data file.
- If you use the /e N command-line option for dataimp, then dataimp imports data only from the tables that are specified in the input file. If you instead use /e Y, then the contents of the entire data file are imported.

To import newly created records from the data file on UNIX

1. Create a text file that will serve as the input file for the Siebel dataimp utility.

   The contents of the file must be formatted as follows (replacing lang_code tokens with actual values that represent the environment):

```
S_LST_OF_VAL
 where LANG_ID in ('lang_code1', ..., 'lang_codeN')
```

2. Save the file with the extension .INP.
3. Navigate to the `siebsrvr/bin` directory.
4. From this directory, execute a command like the following to run the Siebel dataimp utility (replacing $ tokens with actual values that represent the environment):

```
dataimp /u $UserName /p $Password /c "$ODBCDataSource" /d $SiebelTableOwner /f
$DataFileName.dat /h Log /e N /i $InputFileName.inp /A 1
```

5. Check the log file for errors.
6. Connect to the database using a database client that allows you to run SQL interactively, to verify that the imported records are there.

   For example, you execute the following query:

```
select count (LANG_ID), ACTIVE_FLG from S_LST_OF_VAL where LANG_ID = 'lang_code'
group by ACTIVE_FLG;
```

If the result set returns both active and inactive records, then you must investigate and see why some records have been deactivated, and either re-activate or delete these inactive records.

Related Topics

About the Siebel Database Configuration Wizard Utilities

Starting the Siebel Upgrade Wizard

Starting the Siebel Upgrade Wizard

Environments: Development, production test, production.

The Siebel Upgrade Wizard executes the upgrade of the Siebel database. It takes a primary upgrade configuration file as input. This file contains environment information and a driver file name. The Upgrade Wizard executes the steps in the driver file to perform the upgrade.

As the Upgrade Wizard performs the steps in the driver file, it lists the steps in a state log. The state log is located in `siebsrvr/LOG/ process/state`, where process is the upgrade process, for example `upgrep_prod_811` (upgrade from 8.1.1, upgrep process, production test or production environment).

If the Upgrade Wizard encounters an error and exits during an upgrade, then you can restart it after correcting the error. The Upgrade Wizard reads the state log and continues the upgrade from the last successfully completed step.

When you run the Database Configuration Utilities on Windows, they will prompt you if you want to start the Upgrade Wizard. When you run the Database Configuration Utilities on UNIX, you must start the Upgrade Wizard manually.

Requirements for Restarting the Upgrade Wizard

If the Upgrade Wizard stops due to errors, then verify that you have met these requirements before restarting the wizard:

- Carefully review the relevant log files to make sure that your upgrade has completed successfully up to that point.
- Back up your complete set of log files, from the beginning of the process to the point at which it stopped, to another directory.

This backup maintains a complete record of your log files, and prevents your previous log files from being overwritten, which could prevent accurate diagnosis of the reason for the break in the upgrade.

- If you are continuing a previous and incomplete schema upgrade, then do not change the Log Output Directory that you previously selected.
- If problems with your environment prevent the upgrade from restarting, then you must restore the database from the prior base version (the version from which you are upgrading). For example, environment problems might occur when table creation fails due to a database problem (insufficient storage or network problems), which cause subsequent upgrade steps to fail.

If you must restore your database and restart the upgrade, then delete or store the upgrade log files. The files are located in the following directory:

Windows: `SIEBEL_ROOT\log\PROCESS\output`

UNIX: `$SIEBEL_ROOT/log/PROCESS/output`

Also delete the `state.log` file. It is located in the following directory:

Windows: `SIEBEL_ROOT\log\PROCESS\state`

UNIX: `$SIEBEL_ROOT/log/PROCESS/state`

Starting the Upgrade Wizard

Use this procedure to start the Upgrade Wizard. See the following topics to stop the Upgrade Wizard, *Stopping the Upgrade Wizard on Windows* and *Stopping the Upgrade Wizard on UNIX*.

To start the Upgrade Wizard

1. Navigate to the following directory:
   - Windows: `SIEBEL_ROOT\bin`
   - UNIX: `$SIEBEL_ROOT/bin`
2. Enter the following command:
   - Windows: `siebupg /m master_UPGRADEOPTION_ENVIRONMENT_VERSION.ucf`
   - UNIX: `srvrupgwiz /m master_UPGRADEOPTION_ENVIRONMENT_VERSION.ucf`

   In these commands, `UPGRADEOPTION_ENVIRONMENT_VERSION` is the portion of the upgrade configuration file name that lists upgrade process, upgrade environment, and the Siebel CRM release (version) from which you are upgrading. The file is located in `SIEBEL_ROOT\bin` (UNIX: `$SIEBEL_ROOT/bin`).

   Numbers like the following examples are used for the Siebel CRM release portion of the file name:
   - 78
   - 80
   - 811 (SEA)

   The following table lists an example of the file names for an upgrade from Siebel CRM version 8.0.

Upgrade Mode	**File Name**
Development environment upgrep	master_upgrep_dev_80.ucf
Development environment upgphys	master_upgphys_dev_80.ucf
Additive Schema Changes	master_additive_gen.ucf
Prepare for Production	master_prepare_for_production_upgrade.ucf

Upgrade Mode	File Name
Production environment upgrep and upgphys	master_upgrep_prod_80.ucf

3. To begin the upgrade, click OK (Windows) or press Enter (UNIX).

   The Upgrade Wizard will notify you when the upgrade process is complete.

Stopping the Upgrade Wizard on Windows

Do not stop the Upgrade Wizard unless you are confident that an error has occurred, and the Upgrade Wizard or a utility it has called is hanging. Some SQL commands issued by the Upgrade Wizard or by its utilities can take considerable time to complete.

If you are not sure whether the Upgrade Wizard is hanging, then create a service request (SR) on My Oracle Support, or contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.

Stopping the Upgrade Wizard can have varying effects on the RDBMS. Before restarting the Upgrade Wizard, review the RDBMS log files. Run SQL commands as needed to resolve errors found in the RDBMS log files.

To stop the Upgrade Wizard on Windows

- Do one of the following:
  - If the Upgrade Wizard has launched a separate command window in which a utility is running, then close the command window. This terminates the utility and stops the upgrade.
  - In the Upgrade Wizard dialog box, click Cancel. The Upgrade Wizard will exit when the current upgrade step is complete. There might be a delay while the step completes in the RDBMS.

Stopping the Upgrade Wizard on UNIX

Do not stop the Upgrade Wizard unless you are confident that an error has occurred, and the Upgrade Wizard or a utility it has called is hanging. Some SQL commands issued by the Upgrade Wizard or by its utilities can take considerable time to complete.

If you are not sure whether the Upgrade Wizard is hanging, then create a service request (SR) on My Oracle Support, or contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.

Stopping the Upgrade Wizard can have varying effects on the RDBMS. Before restarting the Upgrade Wizard, review the RDBMS log files. Run SQL commands as needed to resolve errors found in the RDBMS log files.

To stop the Upgrade Wizard on UNIX

1. If the Upgrade Wizard has started a utility in a child process, then stop the child process.
2. Exit the shell in which the Upgrade Wizard is running.

3. Locate and stop any orphaned child processes started by the Upgrade Wizard.

After the processes terminate, there might be a delay while the RDBMS executes the issued SQL commands.

Related Topic

About the Siebel Upgrade Wizard and Driver Files

Moving the Siebel Repository Files

Environments: Development and production test.

Before doing the production test environment upgrade, you must copy the upgraded repository definition files (schema.ddl and custrep.dat) from the development environment to the production test environment.

Requirement: If you modified repository objects or schema definitions after completing the development upgrade, then you must regenerate the schema.ddl and custrep.dat files. See *Regenerating the Siebel Repository Definition Files*.

To move the repository files

1. In the development environment, navigate to the following directory:

   Windows: `DBSRVR_ROOT\platform`

   UNIX: `DBSRVR_ROOT/platform`

   In these paths, `platform` is the database platform, for example, `DBSRVR_ROOT\DB2`.
2. Copy the following files:

   custrep.dat

   schema.ddl
3. In the production test environment, put these files in the following location:

   Windows: `DBSRVR_ROOT\Platform`

   UNIX: `DBSRVR_ROOT/Platform`
4. Make a copy of these files, and store them in a safe location.

Running the Siebel Case Insensitivity Wizard

Environments: Development and production test.

Platforms: Windows, UNIX, IBM z/OS.

For Siebel CRM 8.1, query features are enhanced to provide indexes that directly support case and accent insensitive (CIAI) queries on eligible text columns. The Case Insensitivity Wizard configures specified columns for CIAI queries by defining CIAI columns and CIAI indexes in the repository. The wizard also sets the Default Insensitivity property for these columns to DB Case & Accent.

The purpose of the enhanced CIAI features is to improve query effectiveness and performance. Running the Case Insensitivity Wizard is optional.

Overview of What the Case Insensitivity Wizard Does

The Case Insensitivity Wizard performs the following functions in the repository to configure columns to support CIAI queries. No columns or indexes are created in the Siebel database until you synchronize the repository to the Siebel database. The columns you want to configure for CIAI queries are called *base columns*:

- Validates the syntax of all records if an input file is used.
- Validates that all specified tables and columns are eligible for CIAI configuration.
- For each eligible base column, defines a new CIAI column. The CIAI column contains the data in the base column converted to uppercase.
- If you select the Single or Copy All index strategy, then the wizard defines an index on the CIAI column.
- If you select the Copy All index strategy, then it defines a copy of all indexes that have the base column as a key. The new indexes reference the CIAI column instead of the base column.
- Sets the Default Insensitivity property for the base column to DB Case & Accent.
- Sets flags and performs other configuration operations in the repository to support CIAI queries.

The Case Insensitivity Wizard can also be run in a special mode to set the Default Insensitivity property on columns that do not have any indexes defined.

The main purpose of the CIAI query enhancements is to provide indexes that can be used for case insensitive searches. The database does not have to perform table scans to locate records. This allows the database to perform case insensitive searches more quickly.

For example, in S_CONTACT, you configure the column LAST_NAME for CIAI queries. The Case Insensitivity Wizard defines a column called LAST_NAME_CI. When you query for the name Smith, the Object Manager creates a query similar to the following (IBM DB2):

```
SELECT column list FROM S_CONTACT
WHERE LAST_NAME_CI = SMITH
```

CIAI Upgrade Issues for IBM DB2 Users

The Case-Insensitive and Accent-Insensitive search functionality requires changes on database schema levels (new columns, indexes, and triggers). When running the ddlimp utility on IBM DB2 databases, tables that have CIAI columns and triggers are not rebuilt. To resolve this issue, see 553429.1 (Article ID) on My Oracle Support.

Choosing the Correct Repository

If you are upgrading a development environment, then run the Case Insensitivity Wizard on the Siebel Repository. Later in the upgrade process, this repository will be renamed *Prior Customer Repository.*

If you have completed an upgrade of the development environment (upgrep, merge, upgphys), then run the Case Insensitivity Wizard on the Siebel Repository. You then must generate another schema.ddl file and use it to update your production test and production environments. Typically, you run the wizard after an upgrade is complete to revise the configuration of columns you have configured for case insensitive queries.

Running the Case Insensitivity Wizard Using an Input File

Oracle provides a recommended input file for Siebel Business Applications and for Siebel Industry Applications. The input files have a .csv extension and are located in the following directory:

Windows: `$SIEBEL_HOME\objects`

These files list columns that are frequently used for queries and are provided as a recommendation. You can edit these files or create new input files as desired.

Requirements: See *About the Siebel Case Insensitivity Wizard* for information on how the Case Insensitivity Wizard works, eligibility criteria, and how to edit input files.

To run the Case Insensitivity Wizard using an input file

1. Review the input file and verify the following:
   - The syntax for all records is correct.
   - The tables and columns are eligible.
   - The specified configuration options are correct.
   - The configuration defaults are acceptable if any configuration options are omitted.
2. In Siebel Tools, open the repository.
3. Lock the tables listed in the input file.
4. From the Tools menu, choose Utilities, then Case Insensitivity.

   The Case Insensitivity Wizard displays.
5. Select "Administer the columns listed in this file".
6. Click Browse.

   The `$SIEBEL_HOME\objects` directory displays, containing the default .csv input files.
7. Select the desired .csv file, and click Open.
8. In the wizard, click Next.

   The Case Insensitivity Wizard validates the syntax of the file and validates the eligibility of all tables and columns. If the file contains errors or eligibility problems, then the wizard lists the records containing errors. If you continue, then the wizard skips records containing errors or eligibility problems.

   If there are errors, then export the listing to a file, correct any errors in the input file, and restart the Case Insensitivity Wizard. Click Export to export the error listing to a text file.

   If there are no errors, then click Next.

   The Wizard displays the records in the input file.
9. Review the configuration settings, and verify that they are correct.
10. If you want to change any configuration settings, then click Export.

    The Wizard exports the listing to a text file in input-file format.
11. Edit the text file, restart the Case Insensitivity Wizard, and specify the edited text file as the input file.
12. If the configuration settings are correct, then click Next.

    The Wizard displays the changes it will make to repository tables and indexes.

13. If you want to save a record of the changes, then click Export.

    The Wizard writes the changes to a text file.
14. Click Finish.

    The Wizard configures the columns in the repository to support CIAI queries.

Running the Case Insensitivity Wizard by Selecting Columns

This procedure assumes the Object Explorer is in Types mode where objects are displayed hierarchically. An alternate way to perform the procedure is to display Object Explorer in Flat mode. Then choose the Column object and navigate to the desired columns.

Requirements: Review *About the Siebel Case Insensitivity Wizard* for information on how the Case Insensitivity Wizard works, eligibility criteria, and how to edit input files.

Running the Case Insensitivity Wizard by selecting columns

1. In Siebel Tools, open the repository.
2. In the Siebel Tools Object Explorer, select Table.
3. In the Tables list, lock the desired table, and highlight it.
4. In the Object Explorer, select Table, then Column.
5. In the Columns list, select the desired columns.
6. Right-click the highlighted columns, and choose Case Insensitivity from the pop-up menu.

   The Case Insensitivity Wizard validates the eligibility of the selected columns. The Wizard lists any columns that have eligibility errors. If you continue, then the wizard skips columns containing errors.

   Export the error listing to a text file for reference, correct any errors, and restart the Case Insensitivity Wizard. Click Export to export the error listing.

   The Wizard displays the configuration settings it will use to configure the columns.
7. Review the configuration settings and verify they are correct.
8. If you want to change any configuration settings, then click Export.

   The Wizard exports the listing to a text file in input-file format. Edit the text file, then run the Case Insensitivity Wizard and specify the text file as the input file.
9. If the configuration settings are correct, then click Next.

   The Wizard displays the changes it will make to the repository tables and indexes.
10. If you want to save a record of the changes, then click Export.

    The Wizard writes the changes to a text file.
11. Click Finish.

    The Wizard configures the columns in the repository to support CIAI queries.

Configuring CIAI Support for Columns That Do Not Have Indexes Defined

If a column does not have any indexes defined, then the Case Insensitivity Wizard does not create new columns or indexes.

However, you can run the wizard in special mode that changes the column's Default Insensitivity property to DB Case & Accent. The Wizard performs these steps on all eligible columns in the repository. You cannot manually select columns.

Requirements: Review *About the Siebel Case Insensitivity Wizard* for information on how the Case Insensitivity Wizard works, eligibility criteria, and how to edit input files.

To configure columns without indexes

1. In Siebel Tools, open the repository.
2. From the Tools menu, choose Utilities, then Case Insensitivity.

   The Case Insensitivity Wizard displays.
3. Select Enable for all unindexed columns.
4. Click Next.

   The Wizard locates unindexed columns that meet CIAI eligibility criteria. The Wizard then displays a list of tables that must be locked.
5. Click Export to export the list of tables to a text file. Then exit the wizard.
6. Lock all the tables in the text file.
7. Start the Case Insensitivity Wizard again, and select Enable for all unindexed columns.

   The Wizard locates unindexed columns and then displays a page listing how they will be configured.
8. Verify that for all the columns that the method is set to Database, and Index Strategy is set to None.

   When the index strategy is set to None, the wizard does not create a CIAI column or indexes.

   When you click Next, the wizard displays a page listing the repository changes it will make.
9. Verify that for all columns, Default Insensitivity will be changed to DB Case & Accent. Click Finish.

   The Wizard makes the changes to the repository.

Related Topics

About the Siebel Case Insensitivity Wizard

Regenerating the Siebel Repository Definition Files

Applying Siebel Additive Schema Changes

Environments: Production test, production.

Platforms: Windows, UNIX, IBM z/OS.

You can run Database Configuration Utilities in Apply Additive Schema Changes mode. This generates a `schema.additive.sql` script that makes schema changes to the Siebel database schema. You do not have to take the database offline to apply this script.

Applying this script to your production database reduces the number of upgrade steps that must be performed while the production database is offline. This reduces production database downtime during the upgrade.

The recommended method for using this feature is to first test it in the production test environment as part of upgrade tuning. After verifying that `schema.additive.sql`does not adversely affect the operation of the application, you then run the script against the production database.

Requirements for Applying Siebel Additive Schema Changes

The following are requirements for applying Siebel additive schema changes:

- You must have run the Database Configuration Utilities in Apply Additive Schema Changes mode to generate the `schema.additive.sql`script.
- You must have reviewed the Apply Additive Schema Changes log file and verified that `schema.additive.sql` was generated without errors.
- If you are testing the script in the production test environment, then the schema of the Siebel database must be the same as the schema of the production environment Siebel database.

Applying Siebel Additive Schema Changes

Use the following procedure to apply Siebel additive schema changes.

To apply additive schema changes

1. Navigate to the directory where `schema.additive.sql` is located.
2. Review the script and verify that it does not make unacceptable changes to the schema.
3. Production environment only. Compare the script generated in the production environment with the one generated in the production test environment. If the scripts are not identical, then determine why. If you cannot determine that the differences are benign, then do not apply additive schema changes in the production environment.
4. Using any SQL editor, run the script against the Siebel database.

   You can run `schema.additive.sql` multiple times. For example, you can run the script, revise it, and run it again.

Related Topic

About Siebel Additive Schema Changes Mode

Regenerating SQL Files for a Siebel Upgrade

Environments: Production test, production.

If you enter incorrect information in the Database Configuration Utilities, then the SQL files the utilities generate will be incorrect and must not be used. You must run the utilities again, enter the correct information, and regenerate the SQL files.

For example, you are upgrading a non-Unicode database. You run the Database Configuration Utilities and enter that you are upgrading a Unicode database. The SQL that the utility generates will be incorrect and cannot be used. You must regenerate the SQL files.

When the Database Configuration Utilities generate SQL files, all previously generated SQL files and primary UCF files which contains the values of the required parameters are overwritten and newly created.

To regenerate SQL files with the Database Configuration Utilities

1. Verify that the SQL files must be regenerated.
   Typically this is caused by entering incorrect information in the Database Configuration Utilities. If you are unsure if the files must be regenerated, then create a service request (SR) on My Oracle Support, or contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.
2. Rerun the Database Configuration Utilities.

Related Topic

About the Siebel Database Configuration Wizard Utilities

Identifying and Dropping Obsolete Indexes for a Siebel Upgrade

Environments: Production test environment only. Does not apply to production environment.

Use this topic to identify indexes that might be obsolete in the Siebel database and can be dropped (removed). This topic is optional but is recommended since dropping obsolete indexes improves database performance.

When you run the Database Configuration Utilities in Prepare for Production mode, they do the following to identify obsolete indexes:

- Compares the repository schema definition in the development environment against the Siebel database physical schema definition in the production test environment.
- If an index is present in the Siebel database physical schema definition but not in the repository logical schema definition, then the utility creates an SQL drop statement and adds it to that index. The utility places this SQL statement in a file called gen_obs_idx.sql.

You must manually review the gen_obs_idx.sql file. If it contains indexes you want to remove (those containing a drop statement not followed by a create statement), then you must copy the corresponding SQL statements to another SQL file called obs_idx.sql. This file is executed by the Database Configuration Utilities in upgrep mode.

When the Upgrade Wizard is then run, all indexes, including obsolete indexes, are maintained during table rebuilds and data migration. The obsolete indexes file is executed during the Create Siebel Indexes step.

Requirements: You must have run the Database Configuration Utilities in Prepare for Production mode in the production test environment.

To identify and drop (remove) obsolete indexes

1. Navigate to the following file:

   Windows: `DBSRVR_ROOT\platform\gen_obs_idx.sql`

   UNIX: `DBSRVR_ROOT/platform/gen_obs_idx.sql`

   In these paths, platform is the database type, for example DB2.
2. Open the file with a text editor and review the SQL statements it contains.

   The SQL statements drop indexes that are present in the Siebel database but not in the development environment repository logical schema definition.
3. If you want to remove an index, then copy the corresponding SQL statement(s) to the following file:

   Windows: `DBSRVR_ROOT\platform\obs_idx.sql`

   UNIX: `DBSRVR_ROOT/platform/obs_idx.sql`

   In these paths, platform is the database type, for example DB2.

   This file will be executed when you run the Database Configuration Wizard in upgrep mode.

Preparing for a Nondevelopment Environment Siebel Upgrade

Environments: Production test, production.

Platforms: Windows, UNIX, IBM z/OS.

If your installation does not include a development environment, then you do not need to merge your Siebel Repository. Instead, you can use the repository and schema definition files included in the Siebel Database Server installation. The production environment also contains the entire repository.

Before performing the upgrade, you must move and rename these files.

To prepare for a Siebel upgrade in a nondevelopment environment

1. Navigate to `DBSRVR_ROOT \common` (UNIX: `DBSRVR_ROOT /common`) and locate the `mstrep.dat` file.
2. Copy the `mstrep.dat` file and rename it `custrep.dat`.
3. Place the `custrep.dat` file in the `DBSRVR_ROOT\Platform` (UNIX: `DBSRVR_ROOT / platform`) directory, where `platform` is the database platform, for example `DBSRVR_ROOT \DB2`.
4. In the `platform` directory, copy the `ddl.ctl` file and paste the copy into the same directory.
5. Rename the copy `schema.ddl`.
6. In the production test environment create a new database, separate from the Siebel database.

   Install the Siebel database from the new release in the new database. Do not migrate any data to the new database. This database is called the reference database.
7. Define an ODBC for the reference database.

Fixing Column Alignment for Custom Objects

The Fixing Column Alignment for Custom Objects Utility sets the standard for list column header and data alignment based on type of fields they are mapped to across all Siebel applications.

The Fixing Column Alignment for Custom Objects Utility does the following:

- List columns mapped to the Currency field type are aligned right (to the far edge of the column).
- List columns having icon maps mapped to them are center aligned.
- List columns mapped to a Numeric field type are center aligned.
- List columns mapped to a Boolean field (or CHAR(1)) type are center aligned.
- List columns mapped to any other field type are left aligned (to the near edge of the column).
- List columns data alignment will be same as its header alignment.

Left alignment is the default value for data and the header in list columns. HTML Width for list columns mapped to Date field type will be 185 by default.

The Fixing Column Alignment for Custom Objects Utility runs as part of the upgrade process. Use the procedures in this topic to manually run the Fixing Column Alignment for Custom Objects Utility.

Running the Fixing Column Alignment for Custom Objects Utility

1. Navigate to the following directory:

   Windows: `$DbsrvrRoot \common folder`

   UNIX: `$DbsrvrRoot \common folder`
2. Run the following command:

   ```
   java -jar alignapplet.jar /s [Siebel Tool/Server Path, one folder up BIN directory]
   /c [DSN] /d [Database Type - Oracle, DB2UDB, MSSQL or DB2390 ] /u [Database User]
   /p [Database Password] /o [Log directory] /t [Table Owner] /v "Prior
   Customer Repository" /r "New Siebel Repository" /x "New Customer Repository"
   ```

3. Review the AllignApplet log and verify that AlignApplet ran without errors.

   Windows: `$SiebelLogDir \AlignApplet.log`UNIX: `$SiebelLogDir /AlignApplet.log`
4. Review the AllignApplet reports to see the modified objects.

   - **AlignApplet.log.** A log of all the steps executed in this program.
   - **AlignAppletLog.log.** A log folder that contains all the SQL logs executed during the column alignment process.
   - **AlignApplet.html.** A report detailing the modified objects.

Inactivating Unreferenced Repository Objects

The Repository Crawler utility optionally identifies unreferenced Repository objects and inactivates them. The list of objects checked includes:

- Screen
- View
- Applet
- Business Object
- Business Component
- Integration Objects
- Link
- Pick List
- Task Group
- Task

The benefit of running the utility is that having fewer active objects in the Repository will result in a faster Siebel CRM Upgrade and on-going maintenance of your Siebel CRM Enterprise Software.

Note: The process of inactivating Unreferenced Repository Objects can only be run before the upgrade. It cannot be run after the Upgrade.

Running the Inactivating Unreferenced Repository Objects Utility

1. Navigate to the following directory:

   Windows: `$DbsrvrRoot \common folder`

   UNIX: `$DbsrvrRoot \common folder`
2. Run the following command:

```
java -jar RepCrawler.jar /s [Siebel Tool/Server Path, one folder up BIN directory]
/c [DSN] /d [Database Type - Oracle, DB2UDB, MSSQL or DB2390 ] /u [Database User]
/p [Database Password] /o [Log directory] /t [Table Owner] /r "Prior Customer
Repository" /l [logfilename] /j [Custom Script Directory] /a Y /z [DbsrvrRoot\common
path]
```

   RepCrawler refers to the following files (present in the `dbsrvr/common` folder) to determine whether all active objects for active applications are referenced in the code and repository, and subsequently whether to inactivate those objects or not:

   - **RepSanitizationRules.xml.** This *rules* file defines the relationship between objects. The use of object references will be determined by the relationships defined in this file.
   - **RefReferenceObjects.txt.** This *blocklist* file contains a list of all the default (out-of-the-box) objects used by Siebel. Customers can expand this list by adding objects, which they do not want to inactivate as part of the RepCrawler process, to this list.

After RepCrawler runs, any objects which are not referenced in the code and repository will be inactivated.

3. Review the RepCrawler log and verify that RepCrawler ran without errors.
   Windows: `$SiebelLogDir \RepCrawler.log`
   UNIX: `$SiebelLogDir / RepCrawler.log`
4. Review the RepCrawler reports to see the inactive objects.
   - **RepCrawler.html.** A report where you can see the inactive objects.
   - **RepcrawlerLog.log.** A log folder that contains all SQL logs executed during the Repcrawler process.
   - **repcrawler.js.** A data file in JSON format of all inactive objects used in the HTML report.

Converting Siebel Web Templates with the SWT to OD Conversion Utility

The SWT to OD Conversion Utility converts Siebel Web Templates to an Object Definition Layout. Converted Web templates are stored in the database.

The SWT to OD Conversion Utility runs as part of the upgrade process. Use the procedures in this topic to manually run the SWT to OD Conversion Utility. The SWT to OD Conversion Utility can also be run manually if you need to run template migration for a few templates i.e., for importing non-existing custom web templates. However, all templates are migrated during the upgrade process.

Running the SWT to OD Conversion Utility

1. Navigate to the following directory:
   Windows: `$DbsrvrRoot\common folder`
   UNIX: `$DbsrvrRoot\common folder`
2. Run the following command:

```
java -jar $DbsrvrRoot\common\SWTClob.jar /s $SiebelRoot /c "$ODBCDataSource"
/t$TableOwner / u $TbloUser /p $TbloPassword /o $SiebelLogDir /d
$DatabasePlatform/r $RepositoryName /j $WebTemplatesDir /w $WSUSerName /x
$WorkspaceName /b $ParentWS /i $WebTemplateName
```

   Where:
   - $DbsrvrRoot is the path of the database server installation.
   - $SiebelRoot is the path of the Siebel Server installation.
   - $SiebelLogDir is the path to the log directory.
   - $DatabasePlatform is the database platform, such as Oracle, DB2UDB, MSSQL, and DB2390.
   - $RepositoryName is the Name of the Repository, for example, Siebel Repository.
   - $WebTemplatesDir is the directory for Web Templates, for example, $SiebelRoot/Webtempl.
   - $WebTemplateName is the list of template names for the incremental migration.

> **Note:** The physical web template files corresponding to the web template names passed using this parameter should exist in the file system. If not passed, all Web Templates are considered for the incremental migration. Multiple template names should be passed comma separated. for example, /i "Banner,Abcd,Advanced Search".

The following are the optional parameters that are applicable only for the Workspace Environment.

- $WSUserName is the Workspace owner user, for example, SADMIN.
- $WorkspaceName is the new Workspace Name which will be created by the process automatically. for example, dev_xxxx_xxx.
- $ParentWS is the Workspace name under which the $WorkspaceName needs to be created, for example, MAIN.

3. Review the SWTClob log and verify that SWTClob ran without errors.
   Windows: `$SiebelLogDir\SWTClob.log`
   UNIX: `$SiebelLogDir/SWTClob.log`
   For more information about troubleshooting and verifying the results of the Web template migration process, see *Using Siebel Tools* .

Siebel Database Update Process

This topic describes Siebel database update for Siebel CRM, which involves running Post Installation Database Update (that is, the PostInstallDBSetup utility) via Siebel Enterprise Server installer. It includes the following information:

- *Post Installation Database Update*
- *Values Required by Post Installation Database Update*
- *Database Configuration in Siebel Enterprise Server Installer*
- *Troubleshooting Database Configuration*
- *Postpone Running Post Installation Database Update*
- *Skip Post Installation Database Update*
- *Post Installation Database Update Report*
- *Post Installation Database Update Exit Codes*
- *Logging and Diagnostics*
- *Rerunning Post Installation Database Update*

Post Installation Database Update

Post Installation Database Update (the PostInstallDBSetup utility) runs whenever you install the latest monthly update as an update for an existing deployment of Siebel CRM 17.x or later. Where applicable, you run the PostInstallDBSetup utility on every Siebel database (both development and Runtime Repository environments).

> **Note:** Post Installation Database Update is integrated with Siebel Enterprise Server installer and it applies to update installations only; it does not apply if you are installing a new Siebel database or running an Incremental Repository Merge.

The PostInstallDBSetup.zip file, located in the `siebsrvr\bin` folder, contains the payload for PostInstallDBSetup. Unzip the file to review the updates that the PostInstallDBSetup utility will run.

The PostInstallDBSetup utility runs several processes to ensure that the customer database schema, manifest, and seed data is up to date for the current monthly update release.

You can select whether to execute, defer, or skip running the PostInstallDBSetup utility during the update installation, as follows:

- **Execute.** Select this option if you want the installer to collect the database details you provide (see *Database Configuration in Siebel Enterprise Server Installer*) and run the PostInstallDBSetup Utility. This option is selected by default.
- **Defer-Generate DDL files only.** Select this option if you want the installer to collect the database details you provide (see *Database Configuration in Siebel Enterprise Server Installer*) and generate an SQL file containing Data Definition Language scripts for running the PostInstallDBSetup utility manually later.

  If you select this option, the installer does not run the PostInstallDBSetup utility and you must manually run the PostInstallDBSetup utility later after completing the update installation. For more information, see *Postpone Running Post Installation Database Update*.
- **Skip.** Select this option if you do not want the installer to run the PostInstallDBSetup utility and to ignore any data you specify. For example, use this option in installation cases where PostInstallDBSetup is not required. For more information, see *Skip Post Installation Database Update*.

You must provide valid values to the PostInstallDBSetup utility. If the utility does not complete correctly or if you provide invalid values, then the utility exits. Where applicable, you must then run the PostInstallDBSetup utility manually, as described in *Rerunning Post Installation Database Update*.

After the PostInstallDBSetup utility has completed running (see step 3 in *Database Configuration in Siebel Enterprise Server Installer*), a *Post Installation Database Update Report* is automatically generated.

Post Installation Database Update should be run as part of the server installation process, but can also be run manually in situations where you require DBA assistance to make schema changes. For more information, see *Postpone Running Post Installation Database Update*.

Other activities and utilities that Post Installation Database Update is responsible for running include the following:

- **WSRanking utility**. For more information, see *Logging and Diagnostics*.
- **WFUpgrade utility.** For more information, see *Logging and Diagnostics*.
- **TaskUpgrade utility.** For more information, see *Logging and Diagnostics*.
- **REP_VER_NUM creation.** For more information, see *Siebel Repository and Workspace Version Tracking*
- **Workspace enables List of Values.** For more information, see *How PostInstallDBSetup Fully Workspace Enables List of Values*

Note: After applying a Siebel CRM monthly update, the PostInstallDBSetup utility must be successfully run before attempting to use any Siebel CRM component.

Values Required by Post Installation Database Update

During the Siebel Enterprise Server installation process, you must provide database credentials and other information to successfully run the PostInstallDBSetup utility. Siebel Enterprise Server installer will attempt to prepopulate these values from the contents of existing `master_*.ucf` files generated during the original installation or update of Siebel CRM from a previous version. The order that the installer will check these files is as follows:

- The `master_install.ucf` file, which is created during a fresh installation of Siebel CRM.
- The `master_upgrade_dev.ucf` file, which is created during the update of Siebel CRM from a pre-Siebel CRM 17.x version.

If neither file is located or any required information is missing from the files, then you must enter the database information manually into the installer as described in *Database Configuration in Siebel Enterprise Server Installer*.

Database Configuration in Siebel Enterprise Server Installer

In the Siebel Enterprise Server installer for your monthly update release, users (including administrator users) must review and modify the database details when prompted by the installer. The following database configuration in Siebel Enterprise Server installer is required:

1. Select the Edit Configuration check box on the DB Config Utility Screen of the installer to modify the database parameters.
    - The following table describes the parameters that apply for all databases:

Field	Description	Example
RDBMS	Database Platform Name	Oracle Database Enterprise Edition Microsoft SQL Server IBM DB2 UDB for Linux UNIX Windows IBM DB2 for z/OS
Database Table Owner	Table Owner	SIEBEL
Database Table Owner User	Table Owner User	SIEBEL
Database Table Owner Password	Table Owner Password	********
Database Username	Database Username	SADMIN
Database Password	Database Password	********
ODBC DataSource Name	ODBC DataSource Name	Siebel_DSN
Repository Name	Siebel Repository Name	Siebel Repository

Field	Description	Example
Security Group Id/Grantee	Security Group Id/Grantee	SSE_ROLE
UNICODEDB	Y for Unicode DB, N for non-Unicode DB	Y
PRIMARY_LANG_CD	Primary language	English
OTHER_LANG_CD	Other languages. If you have more languages in your database, highlight them here.	German Japanese Hebrew

- The following table describes the parameters that apply for Oracle Database and DB2 UDB:

Field	Description	Example
Index Tablespace Name	The default Indexspace	INDSPC, SEBL_IDX
Tablespace Name	The default Tablespace	TBLSPC, SEBL_TBL

- The following table describes the parameters that apply for IBM DB2 UDB for Linux, UNIX, and Windows:

Field	Description	Example
Tablespace 16K	16K Page Tablespace	TBLSPC16K
Tablespace 32K	32K Page Tablespace	TBLSPC32K

- The following table describes the parameters that apply for DB2390 databases:

Field	Description	Example
4KBUFFERPOOL	4K Buffer Pool (Required).	`BP1`
8KBUFFERPOOL	8K Buffer Pool (Required).	`BP8K1`
16K BUFFERPOOL	16K Buffer Pool (Required).	`BP16K1`

Field	Description	Example
32KBUFFERPOOL	32K Buffer Pool (Required).	BP32K1
DBNAMEPREFIX	Database Name Prefix (Required).	`D + Last 3 Characters of your schema qualifier`
DBCODE	Database Code Page (Required).	`ASCII/EBCDIC/UNICODE`
INDEXBUFFERPOOL	Index Buffer Pool (Required).	`BP2`
STOGROUPINDEXES	Storage Group for Indexes (Required).	`SYSDEFLT`
STOGROUPTABLES	Storage Group for Tables (Required).	`SYSDEFLT`

2. When prompted by the installer, enter the database password which is not stored in any configured files.
3. When installation of the update release completes, the installer calls the PostInstallDBSetup utility and updates the status returned. That is, the installer displays whether the PostInstallDBSetup utility ran successfully or failed.

   The PostInstallDBSetup utility executes the required seed, schema, and manifest changes to the Siebel database. If there are failures with Post Installation Database Update, then re-execute the PostInstallDBSetup utility in standalone mode, after fixing any errors. For more information, see *Rerunning Post Installation Database Update*.

   After the PostInstallDBSetup utility has completed running, a *Post Installation Database Update Report* is automatically generated.

Troubleshooting Database Configuration

Important issues to note before configuring your database for the Siebel CRM monthly update are:

1. On all database platforms, the ODBC Data Source Name that you provide during installation (that is, when prompted by the installer wizard for Siebel Enterprise Server installer) must match the `[Datasources]` and `[ConnectString]` information provided in the siebel.cfg file, located in `C:\siebel\siebsrvr\bin` or equivalent location.

   For example, the following setting during installation:

   ```
   ODBC data source name: Siebel_DSN
   ```

   must match the following in the siebel.cfg file for all database platforms:

   ```
   [DataSources]
   ServerDataSrc = Server
   ```

```
[ServerDataSrc]
ConnectString = siebel_DSN
```

Before installing Siebel CRM monthly updates and in order to ensure successful patch application, you must verify that all respective files are available for your database (MSSQL, Oracle, or DB2UDB & DB2390). In the case of Oracle database, certain changes are also required. For more information, see the following topics:

- *#unique_268/unique_268_Connect_42_section_y5z_34r_mlb*
- *#unique_268/unique_268_Connect_42_section_cjt_g4r_mlb*
- *#unique_268/unique_268_Connect_42_section_r1r_d4r_mlb*

2. For all database types, the CurrentSQLID parameter must be set correctly in the `[ServerDataSrc]` section of the siebel.cfg file.

```
[ServerDataSrc]
CurrentSQLID = SSE_ROLE
```

3. (Specific to Oracle Database) Add the following registry entry in the Windows registry:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\ODBC\ODBC.INI\<Data_Source_Name>]
```

Verify that the following registry keys are set as shown:

```
ColumnsAsChar = 1
ColumnSizeAsCharacter = 1
```

These values are required for the ODBC driver to behave correctly.

4. If execution of the PostInstallDBSetup utility fails, then rerun the PostInstallDBSetup executable using the following command from the command line:

```
PostInstallDBSetup.exe -i Setup.ini -p xxxx -z xxxx
```

For more information, see *Rerunning Post Installation Database Update*.

Verify Files for MSSQL Database

Before installing Siebel CRM monthly updates and in order to ensure successful patch application, you must verify that the following files are available for MSSQL database:

- ODBCAD32.exe
- siebel.cfg

The following table includes more information about these files.

Files	File Location	MSSQL Database
ODBC Datasource	Windows: `C:\Windows\SysWOW64\odbcad32.exe`	Connects to `siebel_DSN`.

Siebel.cfg	Windows:
	`C:\siebel\siebsrvr\bin\enu`
		`[DataSources]`
		`ServerDataSrc = Server`

Files	File Location	MSSQL Database

		`[ServerDataSrc]`
		`ConnectString = siebel_DSN`

Verify Files for Oracle Database

Before installing Siebel CRM monthly updates and in order to ensure successful patch application, you must verify that the following files are available and that all required changes have been made for Oracle database:

- ODBCAD32.exe
- tnsnames.ora
- siebel.cfg

The following table includes more information about these files.

Files	File Location	Oracle Database

tnsnames.ora	Windows:
	
	`<Oracle client location>`
	
	UNIX:
	
	`<Root directory>`
		A **`hostname_instance`** section is created by default in the tnsnames.ora file. Copy and paste the **`hostname_instance`** section and create a new section with, for example, the name siebel_DSN. Note that the siebel_DSN name should be the same across all files for the Oracle database.
		
		`siebel_DSN =`
		`(DESCRIPTION =`
		` (ADDRESS_LIST =`
		` (ADDRESS =`
		` (COMMUNITY = siebel.com)`
		` (PROTOCOL = TCP)`
		` (HOST = hostname)`
		` (PORT = 1551)`
		` )`
		` )`
		` (CONNECT_DATA =`
		` (SID = instance)`
		` )`
		`)`

ODBC Datasource	Windows:
	
	`C:\Windows\SysWOW64\odbcad32.exe`
	
	UNIX:
	
	`/<BuildLocation>/ses/siebsrvr/sys/.odbc.ini`
		Connects to **`siebel_DSN`** on Windows and UNIX.
		
		In the **`[Siebel_DSN]`** section, set ServerName = siebel_DSN, which is defined in tnsnames.ora.
		
		UNIX file content:
		
		`[siebel_DSN] Driver =`
		`/export/home/sblqa/2303/ses/siebsrvr/lib/`
		`SEor827.so`
		`ColumnSizeAsCharacter = 1`
		`ColumnsAsChar = 1`
		`ArraySize = 160000`
		`ServerName = siebel_DSN`

Siebel.cfg	Windows:	In the **`[ServerDataSrc]`** section, set ConnectString = siebel_DSN.

Files	File Location	Oracle Database

	`C:\siebel\siebsrvr\bin\enu`
	
	UNIX:
	
	`/<BuildLocation>/ses/siebsrvr/bin/enu`
		`[DataSources]`
		`ServerDataSrc = Server`
		
		`[ServerDataSrc]`
		`ConnectString = siebel_DSN`

Verify Files for DB2UDB & DB2390

Before installing Siebel CRM monthly updates and in order to ensure successful patch application, you must verify that the following files are available for DB2UDB & DB2390 databases:

- ODBC Datasource
- Siebel.cfg
- db2cli.ini

The following table includes more information about these files.

Files	File Location	DB2UDB & DB2390 Databases

ODBC Datasource	Windows:
	
	`C:\Windows\SysWOW64\odbcad32.exe`
	
	UNIX:
	
	`/<BuildLocation>/ses/siebsrvr/sys/.odbc.ini`
		Connects to `siebel_DSN` on Windows and UNIX with ServerName = <DBNAME>.
		
		UNIX file content:
		
		`[siebel_DSN]`
		`Driver = /export/home/sblqa1/sqllib/lib/libdb2.a`
		`ServerName = <DBNAME>`

Siebel.cfg	Windows:
	
	`C:\siebel\siebsrvr\bin\enu`
	
	UNIX:
	
	`/<BuildLocation>/ses/siebsrvr/bin/enu`
		`[DataSources]`
		`ServerDataSrc = Server`
		
		`[ServerDataSrc]`
		`ConnectString = siebel_DSN`

db2cli.ini	Windows:
	
	`C:\ProgramData\IBM\DB2\<DB2COPY>`
	
	UNIX:
	
	`/<Build location>/ses/siebsrvr`
		Connects to `siebel_DSN` on Windows and UNIX with dbalias=<DBNAME>.
		
		`[siebel_DSN]`
		`dbalias = <DBNAME>`
		`txnisolation = 1`

Postpone Running Post Installation Database Update

In some implementations, all database changes are managed by a DBA team that is distinct from the Siebel CRM team member who is installing the monthly update. This prevents the Siebel CRM installer from successfully running Post Installation Database Update (PostInstallDBSetup utility), since only the DBA team has the required Table Owner credentials.

In this scenario, the strategy is to let Post Installation Database Update generate the required Data Definition Language (DDL) scripts to make schema changes, which can then be edited and applied by the DBA team later. After this has occurred, the Siebel CRM installer can manually run Post Installation Database Update to create the required seed and manifest data (it will not need to make any schema changes).

If you want to postpone running the PostInstallDBSetup utility during a monthly update release, then complete the steps in the following procedure.

To postpone Post Installation Database Update

1. In the Siebel Enterprise Server installer for your monthly update release, select the *Defer-Generate DDL files only* option to postpone running the PostInstalDBSetup utility.
   The installer collects the database details you provide and generates an SQL file containing Data Definition Language scripts for manually running the PostInstallDBSetup utility later (see step 3), after completing the update installation.
2. To apply the schema changes to the database, the DBA must execute `generate_schema.dll`, which is located in the following folder:

   > **Note:** The DBA can modify the `generate_schema.dll` file to specify environment-specific parameters, such as table or index spaces.

   ```
   ses\siebsrvr\log\PostInstallDBSetup_<time_stamp>\Common
   ```
3. Manually run the PostInstallDBSetup utility to create the remaining artifacts in the database. That is, the required seed and manifest data; there are no schema changes since you manually applied the schema changes in step 2.
   Before running PostInstallDBSetup manually, modify the setup.ini file (located in the `..\ses\siebsrvr\bin` folder) to indicate that the schema has already been installed by setting the following parameter:
   ```
   generate_schema=A
   ```
   For more information, see *Rerunning Post Installation Database Update*.

Skip Post Installation Database Update

If you do not want the installer to run the PostInstallDBSetup utility, then complete the steps in the following procedure. Post Installation Database Update is required in all situations except the following:

- You are installing a new Siebel database instance that does not exist anywhere.
- You have an existing instance of a pre-Siebel CRM 17.x version that you want to run an update or Incremental Repository Merge against after installing the binaries.

- You have already run Post Installation Database Update during a previous Siebel Enterprise Component update for the same monthly update in the same enterprise.

To skip Post Installation Database Update

- In the Siebel Enterprise Server installer for your monthly update release, select the *Skip* option to skip running the PostInstalDBSetup utility.

  The installer will not run the PostInstallDBSetup utility.

Post Installation Database Update Report

A report called `PostInstallDBSetup_<Timestamp>.html` is automatically generated after you run the PostInstallDBSetup utility. This report is in HTML format, is located in `siebsrvr\log\PostInstallDBSetup_<Timestamp>.html`, and provides information about the update including the following:

- The objects that are delivered out-of-the-box.
- The seed, schema, and manifest data that was imported as part of the database update process for a monthly update.
- Any area or step where the PostInstallDBSetup utility fails.

To review the Post Installation Database Update Report

1. Go to the following location on your local drive: `siebsrvr\log\PostInstallDBSetup_xxxxxxxxx`
2. Double click `PostInstallDBSetup_<Timestamp>.html` to open the report you want to view, where `<Timestamp>` represents the date and time that the report was generated and is in the following format: YYYY-MM-DD_HR-MIN-SEC.
3. In the Post Installation Database Setup Report console that opens:

   - Click Upgrade Summary to review summary information about all the steps that were run as part of the Post Installation Database Update process. Summary information is available for the following: Parameters, Initialization, WSRanking, SeedSchemaManifest Upgrade, and WorkflowUpgrade. Subcategories that appear under SeedSchemaManifest Upgrade include the following:

     - **Initialization.** Shows summary information related to initialization tasks and the status of each step (Completed/Failed/Error) in the process. Initialization tasks include validation of parameters, customer's database version, and database connectivity.
     - **Schema.** Shows summary information related to schema import and the status of each step (Completed/Failed/Error) in the process.
     - **Manifest.** Shows summary information related to schema and data import and the status of each step (Completed/Failed/Error) in the process.
     - **Seed.** Shows summary information related to data import and the status of each step (Completed/Failed/Error) in the process.
   - Click Schema to review the names of the tables and columns that were updated out-of-the-box.
   - Click Manifest to review the basic user key information for all `S_UI*` tables.
   - Click Seed to review all the tables (and user key information) for which seed data was released.
   - Click Troubleshoot to review and troubleshoot errors that occur during Post Installation Database Update.

Post Installation Database Update Exit Codes

The following table describes all Post Installation Database Update (PostInstallDBSetup utility) exit codes.

Return Code	Description (Internal Only)	Console Message
0	WSRanking - Successful SeedSchema - Successful WFUpgrade - Successful TaskUpgrade - Successful	Successfully completed WSRanking Utility Execution Successfully completed SeedSchemaSetup Utility Execution Successfully completed Workflow Upgrade Utility Execution Successfully completed Task Upgrade Utility Execution Post Installation Database Setup execution Completed
1	WSRanking - NA SeedSchema - Successful WFUpgrade - Successful TaskUpgrade - Successful	WSRanking Utility Execution is Not Applicable Successfully completed SeedSchemaSetup Utility Execution Successfully completed Workflow Upgrade Utility Execution Successfully completed Task Upgrade Utility Execution Post Installation Database Setup execution Completed
2	WSRanking - NA (its RR environment) SeedSchema - NA WFUpgrade - NA TaskUpgrade - NA	RR environments only - none of these utilities are required.
3	WSRanking - Failed SeedSchema - Not run WFUpgrade - Not run TaskUpgrade - Not run	WSRanking Utility Execution failed Post Installation Database Setup execution failed
4	WSRanking - Successful SeedSchema - Failed WFUpgrade - Not run TaskUpgrade - Not run	Successfully completed WSRanking Utility Execution SeedSchemaSetup Utility Execution failed Post Installation Database Setup execution failed
5	WSRanking - NA SeedSchema - Failed WFUpgrade - Not run	WSRanking Utility Execution is Not Applicable SeedSchemaSetup Utility Execution failed Post Installation Database Setup execution failed

Return Code	Description (Internal Only)	Console Message
	TaskUpgrade - Not run	
6	WSRanking - Unknown SeedSchema - Unknown WFUpgrade - Unknown TaskUpgrade - Unknown	Unexpected errors or warnings - review log files.
7	WSRanking - NA SeedSchema - NA WFUpgrade - NA TaskUpgrade - NA	Attempt was made to run against a pre-IP2017 Repository.
8	WSRanking - NA SeedSchema - NA WFUpgrade - NA TaskUpgrade - NA	Attempt was made to run PostInstallDBSetup for a second time for the same version.
9	WSRanking - Successful SeedSchema - Successful WFUpgrade - Failed TaskUpgrade - Not run	Successfully completed WSRanking Utility Execution Successfully completed SeedSchemaSetup Utility Execution Error while Workflow upgrade Utility Execution Post Installation Database Setup execution Completed
10	WSRanking - Successful SeedSchema - Successful WFUpgrade - NA TaskUpgrade - NA	Successfully completed WSRanking Utility Execution Successfully completed SeedSchemaSetup Utility Execution Workflow Upgrade Utility Execution is not Applicable Task Upgrade Utility Execution is not Applicable Post Installation Database Setup execution Completed
11	WSRanking - NA SeedSchema - Successful WFUpgrade - Failed TaskUpgrade - Not run	WSRanking Utility Execution is Not Applicable Successfully completed SeedSchemaSetup Utility Execution Error while Workflow upgrade Utility Execution Post Installation Database Setup execution Completed
12	WSRanking - NA SeedSchema - Successful	WSRanking Utility Execution is Not Applicable Successfully completed SeedSchemaSetup Utility Execution

Return Code	Description (Internal Only)	Console Message
	WFUpgrade - NA TaskUpgrade - NA	Workflow Upgrade Utility Execution is not Applicable Task Upgrade Utility Execution is not Applicable Post Installation Database Setup execution Completed
13	WSRanking - NA SeedSchema - NA WFUpgrade - NA TaskUpgrade - NA	**Defer** option only—indicates that Schema Generation was successful.
14	WSRanking - NA or Successful SeedSchema - NA or Successful WFUpgrade - NA or Successful TaskUpgrade - Failed	NA
15	WSRanking - NA SeedSchema - Successful WFUpgrade - Successful TaskUpgrade - NA	NA
16	WSRanking - Successful SeedSchema - Successful WFUpgrade - Successful TaskUpgrade - NA	NA

The console outputs a full statement for each of the sub-processes invoked by PostInstallDBSetup. These are in one of the following formats:

- <Process> is Not Applicable (for example, “WSRanking Utility Execution is Not Applicable”)
- <Process> Successfully completed (for example, “SeedSchemaSetup Utility Successfully completed”
- <Process> Failed (for example, “TaskUpgrade Utility Execution Failed”)

An overall assessment is added after all the sub-processes have been listed stating “PostInstallation Database Setup Failed/Succeeded”.

Logging and Diagnostics

Review any logging and diagnostic information in the following log files: WSRanking, WFUpgrade, and TaskUpgrade.

WSRanking Logs

To review detailed logs for the WSRanking utility, see the WSRanking_xxxxxxxxx subdirectory in the following (or equivalent) folder location:

```
...\ses\siebsrvr\log\PostInstallDBSetup_xxxxxxxxx
```

The Workspace Ranking utility makes a minor schema update to increase the column length of one of the Workspaces related columns (WS_OBJ_VER).

WFUpgrade Logs

To review detailed logs for the WFUpgrade utility, see the WorkflowUpgrade.log file in the following (or equivalent) folder location:

```
...\ses\siebsrvr\log\PostInstallDBSetup_xxxxxxxxx
```

The Workflow Upgrade utility runs in both development (Design Time Repository or DR) and production (Runtime Repository or RR) environments. The utility generates RR definitions for Workflow Processes, if it has not previously been executed.

- **Workflow DR Upgrade.** There may be multiple versions of the same workflow in different Workspaces; however, only one version of a workflow can be used at runtime in the DR environment. Each workflow must have a corresponding S_RR_WORKFLOW record. The WFUpgrade utility takes care of this by generating RR definition records in S_RR_WORKFLOW for each workflow version in MAIN and Integration branches.
- **Workflow RR Upgrade.** In RR environments, RR definitions for workflows are generated from the S_WFA_DPLOY_DEF table.

TaskUpgrade Logs

To review detailed log for the TaskUpgrade utility, see the TaskUpgrade.log file in the following (or equivalent) folder location:

```
...\ses\siebsrvr\log\PostInstallDBSetup_xxxxxxxxx
```

The Task Upgrade utility runs in both development (DR) and production (RR) environments. The utility generates RR definitions for Tasks, if it has not previously been executed.

- **Task DR Upgrade.** There may be multiple versions of the same task in different Workspaces; however, only one version of a task can be used at runtime in the DR environment. Each task must have a corresponding S_RR_TASK record. The TaskUpgrade utility takes care of this by generating Runtime Repository definition records in S_RR_TASK for each task version in the MAIN and Integration branches.
- **Task RR Upgrade.** In RR environments, RR definitions for tasks are generated from the S_TU_TASK table.

Rerunning Post Installation Database Update

If there are failures with Post Installation Database Update, then re-execute the PostInstallDBSetup utility in stand-alone mode, after fixing any errors, as shown in the following procedure.

To rerun Post Installation Database Update

1. If required, update the setup.ini file in `c:\siebel\siebsrvr\bin` or equivalent location.

   The setup.ini file contains, for example, the following parameter definitions:

   ```
   GENERATE_SCHEMA=N
   SIEBSRVR_ROOT=d:\Siebel\ses\siebsrvr
   REPOSITORY=Siebel Repository
   DBTYPE=mssql
   ODBC_DSN=siebel_dsn
   SIEBEL_DSN=ServerDataSrc
   TBLO=dbo
   TBLOUSER=SIEBEL
   SIEBUSER=SADMIN
   SSE_ROLE=SSE_ROLE
   UNICODEDB=Y
   PRIMARY_LANG_CD=ENU
   OTHER_LANG_CD=TRK,THA,SVE,RUS,PTG,PTB,PLK,NLD,KOR,JPN,ITA,HEB,FRA,FIN,ESN,DEU,DAN,CSY,CHT,CHS,ARA
   ```

2. Rerun PostInstallDBSetup.exe, located in `c:\siebel\siebsrvr\bin` or equivalent location, using the following command from the command line:

   ```
   PostInstallDBSetup.exe -i Setup.ini -p ******** -z ********
   ```

   where:

   - `-i` is the ini file name.
   - `-p` is the Database Table Owner Password.
   - `-z` is the Database Password.

RepositoryUpgrade Utility

The RepositoryUpgrade utility delivers optional repository changes made by the Oracle Siebel CRM development team since Siebel CRM 17.0. The RepositoryUpgrade.zip file, located in the `siebsrvr\bin` folder, contains the payload for RepositoryUpgrade. Unzip the file to see its contents – a subfolder will exist for each release that has changes (for example: 20.3, 20.7, 20.8, ... 21.12, 22.1, and so on).

The RepositoryUpgrade utility inspects the database table S_APP_HIST to determine the last time it ran and the next release folder to start running updates from.

During execution, the RepositoryUpgrade utility imports data from the folders added since the last time it was run. This prevents customers having to re-resolve any conflicts they have already addressed.

If there are 10 folders for RepositoryUpgrade to process, then they will be processed in order of release but only a single Integration Workspace will be created. This ensures that if Object_X changed in 20.7 and again in 21.12, then the 21.12 changes will be merged with the 20.7 changes so that customers get only the latest version.

Assuming it is January 2022, the typical order of events to follow when planning to run the RepositoryUpgrade utility are:

1. Customer identifies a feature that they want in a monthly update (22.1 for example), which requires repository changes.

2. Assuming that the customer performs monthly internal releases, the customer analyzes the effort involved in consuming the changes (merge with existing customizations, test, and so on) and picks a future release when they plan to actually deploy it (April 2022 for example).
3. Customer runs the RepositoryUpgrade utility, pointing to the int_2022_April release branch, where they are staging the changes to be released in April. This keeps all the changes out of the in-progress February and March releases and:
    a. Creates a child Integration Workspace named int_siebel_updateN (where N is a sequential number).

       You cannot have two Workspaces with the same name and you may have to run RepositoryUpgrade more than once for features released in different monthly updates.
    b. Creates a child Developer Workspace under int_siebel_updateN where all changes are imported – you cannot directly edit an Integration Workspace.

       You must review the Developer Workspace to determine what objects have changed and resolve any conflicts before delivering it to the parent int_siebel_updateN branch. See the *Using Siebel Tools* guide for more information about this process.
4. Customer delivers int_siebel_updateN to int_2022_April.

   This will work if there are no customizations to any of the changed objects; otherwise a Rebase will be required and the customer will have to resolve conflicts before delivery.
5. After delivery, int_2022_April will contain the latest (Oracle and all customer) changes and can be migrated to test environments as needed.

Running RepositoryUpgrade Utility

You can *optionally* run the RepositoryUpgrade utility on the development database to install new features since Siebel CRM 17.0 that require repository updates. This task does not apply if you do not require any of these new features.

For information about the new features that require repository updates, see *Siebel CRM Update Guide and Release Notes* on My Oracle Support for your Siebel CRM 22.x Update release.

The following procedure describes how to update the repository by running the RepositoryUpgrade utility, which is located in the `siebsrvr\bin` folder.

Note: Always run the RepositoryUpgrade utility from the Siebel Server home directory (`$SIEBEL_HOME`).

To run RepositoryUpgrade utility

1. Run RepositoryUpgrade (RepositoryUpgrade.exe for Windows or RepositoryUpgrade.sh for other platforms) without any arguments to review the available command line arguments. The following tables describe the returned arguments and parameters, which you can use to run the RepositoryUpgrade utility.

Parameter	Description	Example
`-s`	Siebsrvr installation path (Required)	`C:\$SIEBEL_HOME\ses\siebsrvr`
`-t`	Siebel Table Owner (Required)	`SIEBEL`
`-u`	TBLO Username (Required)	`SIEBEL`

Parameter	Description	Example
`-p`	TBLO Password (Required)	`********`
`-o`	ODBC Data Source (Required)	`Siebel_DSN`
`-d`	Database platform (one of: Oracle, MSSQL, DB2UDB, or DB2390)	`Oracle`
`-y`	Siebel Username (Required)	`SADMIN`
`-z`	Siebel User Password (Required)	`********`
`-r`	Repository Name (Default: "Siebel Repository") (Required)	`Siebel Repository`
`-g`	Primary Base Language	`ENU`
`-w`	Additional Languages (Default: None)	`ARA,CHS,FIN,ITA`
`-c`	SSE_ROLE (Required)	`SSE_ROLE`
`-m`	Seed inp (data input) creation for the Migration Application (Default: Siebel Log Directory) **Note:** The RepositoryUpgrade utility will create a file named datamig.inp, which is required for the Migration Application to migrate seed data related to the repository changes from the DR to RR environment.	`C:\$SIEBEL_HOME\ses\siebsrvr \log`
`-i`	Unicode DB (Default: Y)	`Y`
`-6`	Generate Schema File Only (Default: N) Valid options are: ◦ **Y** (Yes, generate schema file only) – This option generates a file with the required schema changes without actually applying them. This allows your database administrator to apply the schema changes to the physical database outside of RepositoryUpgrade. ◦ **A** (Already applied Schema changes) – This option means schema changes are already applied. Use this option after the database administrator has completed the schema changes in the physical database.	`N`

Parameter	Description	Example
	◦ **N** (No, don't only generate schema file) – This option means that you run RepositoryUpgrade to perform schema changes and import all other changes in one execution. **Note:** Use this switch to generate the required schema changes without actually applying them, thereby allowing your DBA to apply the schema changes outside of RepositoryUpgrade. For more information, see *Independently Apply RepositoryUpgrade Schema Changes* .	
`-9`	Parent Workspace Name **Note:** Best practice is to specify an Integration Workspace reflecting a planned future release (you will receive an error if you specify "MAIN"). This will allow you to merge the Oracle changes with your custom changes and test them before migrating them to Production.	`<Parent_IWS_Name>`
`-b`	Default Tablespace **Note:** Mandatory argument for Oracle, DB2 UDB, and DB2390 databases.	`SEBL_DATA`
`-x`	Default Indexspace **Note:** Mandatory argument for Oracle and DB2 UDB databases.	`SEBL_INDX`
`-k`	16K Page Tablespace **Note:** Mandatory argument for DB2 UDB databases.	`TBLSPC16K`
`-v`	32K Page Tablespace **Note:** Mandatory argument for DB2 UDB databases.	`TBLSPC32K`
`-4`	Provide a suffix for the log folder generated by RepositoryUpgrade. (Default: `Null`) Example: `-4 Ora_23_11` will cause the log folder to be created as `RepositoryUpgrade_Ora_23_11`	`-4 Ora_23_11`

Parameter	Description	Example

The following arguments are required only for DB2390 databases:

Parameter	Description	Example
`-2`	4K Buffer Pool (Required)	BP1
`-5`	8K Buffer Pool (Required)	BP8K1
`-f`	16K Buffer Pool (Required)	BP16K1
`-7`	32K Buffer Pool (Required)	BP32K1
`-8`	Database Name Prefix (Required)	D + Last 3 Characters of your schema qualifier
`-a`	Database Code Page (Required)	ASCII/EBCDIC/UNICODE
`-j`	Index Buffer Pool (Required)	BP2
`-e`	Storage Group for Indexes (Required)	SYSDEFLT
`-h`	Storage Group for Tables (Required)	SYSDEFLT

Note: Don't use parameters that are inappropriate for your database platform. For example, don't use the DB2390 `-j` argument with Oracle, MSSQL, or DB2UDB databases.

2. Execute the RepositoryUpgrade utility using values appropriate to your database. Examples for each supported database platform follow.

   Oracle example:

```
RepositoryUpgrade.exe -s C:\$SIEBEL_HOME -t ORAXXXX -u ORAXXXX -p ******** -o Siebel _DSN
-d ORACLE -y SADMIN -z ******** -b SEBL_DATA -x SEBL_INDX -c SSE_ROLE -9 int_1
```

   MSSQL example:

```
RepositoryUpgrade.exe -s C:\$SIEBEL_HOME -t dbo -u MSXXXX -p ******** -o Siebel_DSN
```

```
-d MSSQL -y SADMIN -z ******** -c SSE_ROLE -9 int_1
```

DB2 UDB example:

```
RepositoryUpgrade.exe -s C:\$SIEBEL_HOME -t SIEBEL -u SIEBEL -p ******** -o Siebel_DSN
-d DB2UDB -y SADMIN -z ******** -b TBS_4K -x TBS_4K -k TBS_16K -v TBS_32K -c SSE_ROLE -9 int_1
```

DB2390 example:

```
RepositoryUpgrade.exe -s C:\$SIEBEL_HOME -t CQ10D0XX -u QADMIN -p XXXXX -o Siebel_DSN
-d DB2390 -y QADMIN -z XXXXX -c SSEROLE -b SIEBTS00 -2 BP1 -5 BP8K1 -f BP16K1 -7 BP32K1
-8 D003 -a ASCII -j BP2 -e SYSDEFLT -h SYSDEFLT -9 int_1
```

3. Review the returned output. Output similar to the following is expected:

```
------------------------------------------
Current Content Version : 21.3
Current DB Version : 19.12
------------------------------------------

------------------------------------------
Following builds available in the installer
------------------------------------------
20.3
20.7
...
21.3
------------------------------------------

------------------------------------------
Following builds need to be installed
------------------------------------------
20.3
...
...
21.3
------------------------------------------

------------------------------------------
Schema Installation Starts
------------------------------------------
Starting for each build
Running for 20.3
...
Running for 21.1
Skipping: No Schema present for 21.2
Running for 21.3
Completed for each build
Publishing Table Starts
Sync Logical and Physical Schema
Publishing Table Completed
------------------------------------------
Schema Installation Completed
------------------------------------------

------------------------------------------
Repository Import Starts
------------------------------------------
Starting for each build
Running for 20.3
...
Skipping: No Repository present for 21.2
```

```
Running for 21.3
Completed for each build
Deliver Workspace Starts
Deliver Workspace Completed
------------------------------------------
Repository Import Completed
------------------------------------------

------------------------------------------
Seed Import Starts
------------------------------------------
Starting for each build
Running for 20.3
...
Running for 21.2
Skipping: No Seed present for 21.3
Completed for each build
Running Seed Copy for LOV for each branch
------------------------------------------
Seed Import Completed
------------------------------------------

------------------------------------------
Manifest Import Starts
------------------------------------------
Starting for each build
Running for 20.3
Skipping: No Manifest present for 20.7
Running for 20.8
Running for 20.9
Skipping: No Manifest present for 20.10
Skipping: No Manifest present for 20.11
Running for 20.12
Running for 21.1
Skipping: No Manifest present for 21.2
Skipping: No Manifest present for 21.3
Completed for each build
------------------------------------------
Manifest Import Completed
------------------------------------------

*** Integration Branch : int_siebel_update ***
*** Workspace : dev_sadmin_siebel_21_3 ***
*** Please review Web Tools -> Archive -> SIF Attribute Differences and resolve any
conflicts. Once done, click Apply Changes to apply your decisions. If you do not make any
changes in the report, all incoming configuration will be used. Once you checkpoint or
version the Workspace, the Apply Changes option will be disabled. ***

RepositoryUpgrade Execution Report Location : C:\$SIEBEL_HOME\ses\siebsrvr\log
\RepositoryUpgrade_20210204_044026.html
```

Check the HTML file (RepositoryUpgrade Execution Report) for more details, resolve any issues, and then rerun the process if required.

At this point, RepositoryUpgrade will have created two new Workspaces in the repository, as follows:

- An Integration Workspace named int_siebel_update_X directly underneath the Workspace specified by the `-9` parameter (provided on the RepositoryUpgrade command line).

- A Developer Workspace under this Integration Workspace named "dev_sadmin_siebel_mm_dd", where mm_dd represents the release you are importing, such as 23.10.

  Please review Web Tools -> Archive -> SIF Attribute Differences and resolve any conflicts. Once done, click Apply Changes to apply your decisions. If you do not make any changes in the report, all incoming configuration will be used. Once you checkpoint or version the Workspace, the **Apply Changes** option will be disabled.

  The Developer Workspace was used to import the Repository changes. This Workspace can be used to inspect all the RepositoryUpgrade-supplied changes and resolve any conflicts before delivery to the parent Integration Workspace. For more information on this, review the section *Viewing and Resolving Conflicts after Siebel Archive File Imports in Web Tools* in the *Using Siebel Tools* guide.

  After completing the conflict resolution, deliver the Developer Workspace to its parent Integration Workspace.

4. Rebase the int_siebel_update_X branch to its parent Integration Workspace and test it in conjunction with your customer changes.
5. Once you are comfortable that your changes and the RepositoryUpgrade-supplied changes work together, deliver the int_siebel_update_X branch to its parent Integration Workspace.

   Depending on your specific implementation and any conflicts that may exist, this process may require a Rebase, Conflict Resolution, Submission for Delivery, approvals in Siebel Approval Manager, and so on. For more information on delivering changes from a Workspace, see the *Using Siebel Tools* guide.

Independently Apply RepositoryUpgrade Schema Changes

If your DBA would like to manage RepositoryUpgrade schema changes, you can instruct RepositoryUpgrade to generate a Data Definition File which the DBA can then use to apply the schema changes external to Siebel CRM utilities.

To independently apply RepositoryUpgrade schema changes

1. Run RepositoryUpgrade with the `-6` argument set to `Y`. This will generate the schema DDL file containing the required database changes, but will not apply them.

   Oracle example:

```
RepositoryUpgrade.exe -s C:\$SIEBEL_HOME -t SIEBEL -u SIEBEL -p ******** -o SIEBEL
-d ORACLE -y SADMIN -z ******** -b SEBL_DATA -x SEBL_INDX -c SSE_ROLE -6 Y -9 int_future_release_X
```

   MSSQL example:

```
RepositoryUpgrade.exe -s C:\$SIEBEL_HOME -t dbo -u SIEBEL -p ******** -o Siebel_DSN
-d MSSQL -y SADMIN -z ******** -c SSE_ROLE -6 Y -9 int_future_release_X
```

   DB2 UDB example:

```
RepositoryUpgrade.exe -s C:\$SIEBEL_HOME -t SIEBEL -u SIEBEL -p ******** -o DB2KLXXX
-d DB2UDB -y SADMIN -z ******** -b TBS_4K -x TBS_4K -k TBS_16K -v TBS_32K -c SSE_ROLE -6 Y -9
 int_future_release_X
```

   DB2390 example:

```
RepositoryUpgrade.exe -s C:\$SIEBEL_HOME -t CQ10D0XX -u QADMIN -p XXXXX -o Q10D
-d DB2390 -y QADMIN -z XXXXX -c SSEROLE -b SIEBTS00 -2 BP1 -5 BP8K1 -f BP16K1 -7 BP32K1
-8 D003 -a ASCII -j BP2 -e SYSDEFLT -h SYSDEFLT -6 Y -9 int_future_release_X
```

2. After execution, provide the `generate_schema.ddl` file to the database administrator, who can inspect the contents and make any required changes, such as specifying alternate Tablespaces or other storage options, and will then execute the file against the database to update the schema.

   ```
   ses\siebsrvr\log\RepositoryUpgrade_<time_stamp>
   ```

3. After the DBA has successfully applied the physical schema changes, rerun RepositoryUpgrade with the `-6` flag set to `A` (indicating that the schema changes have already been applied), as shown in the following example:

   Oracle example:

   ```
   RepositoryUpgrade.exe -s C:\$SIEBEL_HOME -t SIEBEL -u SIEBEL -p ******** -o SIEBEL
   -d ORACLE -y SADMIN -z ******** -b SEBL_DATA -x SEBL_INDX -c SSE_ROLE -6 A -9 int_future_release_X
   ```

   MSSQL example:

   ```
   RepositoryUpgrade.exe -s C:\$SIEBEL_HOME -t dbo -u SIEBEL -p ******** -o Siebel_DSN
   -d MSSQL -y SADMIN -z ******** -c SSE_ROLE -6 A -9 int_future_release_X
   ```

   DB2 UDB example:

   ```
   RepositoryUpgrade.exe -s C:\$SIEBEL_HOME -t SIEBEL -u SIEBEL -p ******** -o DB2KLXXX
   -d DB2UDB -y SADMIN -z ******** -b TBS_4K -x TBS_4K -k TBS_16K -v TBS_32K -c SSE_ROLE -6 A -9
    int_future_release_X
   ```

   DB2390 example:

   ```
   RepositoryUpgrade.exe -s C:\$SIEBEL_HOME -t CQ10D0XX -u QADMIN -p XXXXX -o Q10D
   -d DB2390 -y QADMIN -z XXXXX -c SSEROLE -b SIEBTS00 -2 BP1 -5 BP8K1 -f BP16K1 -7 BP32K1
   -8 D003 -a ASCII -j BP2 -e SYSDEFLT -h SYSDEFLT -6 A -9 int_future_release_X
   ```

4. After successfully executing RepositoryUpgrade, follow the instructions in *Running RepositoryUpgrade Utility* to deliver the Oracle-supplied repository changes to your customer repository.

Siebel Repository and Workspace Version Tracking

Because running the RepositoryUpgrade utility is optional, customers who have not executed the utility may end up in a situation where their Siebel CRM binaries (having installed the latest monthly update) might attempt to draw on repository content that is not actually in their own repository, since they have not run the RepositoryUpgrade utility.

To address this issue, the REP_VER_NUM column has been added to the S_WS_VERSION table so that the current Siebel Repository version (populated by running the RepositoryUpgrade utility) will be linked to a given Workspace version. The association between Siebel Repository version and Workspace version helps determine if the RepositoryUpgrade utility has been executed against a given Workspace version or not.

The S_WS_VERSION table contains a row for each version of every Workspace (MAIN, Integration, or Developer Workspace).

When an object manager starts, the repository version that the application is running against will be logged in the application object manager log. This (repository version) information will assist in debugging unexpected behavior after applying a monthly update.

> **Note:** In Siebel CRM 21.10 Update, the REP_VER_NUM column (predefaulted to 17.0) is automatically added to the S_WS_VERSION table during Post Installation Database Update.

Example

Consider the following example, which shows what happens when a customer runs the RepositoryUpgrade utility for a monthly update and what happens when a customer does not run the RepositoryUpgrade utility.

1. Customer Z previously ran the RepositoryUpgrade utility for 20.9.
2. Customer Z installs the latest monthly update for 21.10 but does *not* run the RepositoryUpgrade utility.

   During Post Installation Database Update, the REP_VER_NUM column is automatically added to Customer Z's S_WS_VERSION table and predefaults to 17.0. In addition, all existing Workspace branches default to 17.0 (since there is no way to determine what has previously been imported, 17.0 is the one thing we can be sure of).
3. As a result of the previous steps and *not* running the RepositoryUpgrade utility:
   - Customer Z's *binary* version is at 21.10 but their *repository* version is at 20.9.
   - Features or bug fixes in 21.10 will not work as expected because the supported repository changes are not present in Customer Z's repository.
   - Due to the time-lapse between 20.9 and 21.10, Customer Z may not be sure if (or when) the RepositoryUpgrade utility was run. For this reason, debugging unexpected behavior may be very difficult.
4. When Customer Z runs the RepositoryUpgrade utility, they specify a particular parent Workspace under which to create a new Integration Workspace (IWS) that will include all the repository changes available in that release – changes which are cumulative from Siebel CRM 17.0 onwards.
   - If Customer Z runs RepositoryUpgrade for 21.10, then 21.10 will be set in `S_WS_VERSION.REP_VER_NUM` for the Integration Branch that RepositoryUpgrade creates.
   - Immediately after running RepositoryUpgrade, 21.10 will only exist in that Integration Branch but after testing the changes in that branch, Customer Z eventually delivers it to the parent IWS. At this stage:
     - The changes from 21.10 will also be in that parent IWS so the S_WS_VERSION record for the parent IWS will be updated to the version from the child that was delivered; *and*
     - Then that version will work its way upwards in the Workspace hierarchy until it ends at MAIN.
   - MAIN, at this point, and the children that were delivered will have the repository changes provided in the monthly update.

Related Topics

- Setting the RepVer Column Compatibility System Preference for DB2 Database

11 Siebel CRM Upgrade Factory

Siebel CRM Upgrade Factory

This chapter provides information on the following topics:

Overview of the Siebel CRM Upgrade Factory

Siebel CRM Upgrade Factory delivers an automated development upgrade that includes the quick setup of a development environment. The upgrade factory approach simplifies the application upgrade process, allowing customers to upload their customized, pre-IP2017 repository to run an upgrade and IRM process on Oracle Cloud Infrastructure (OCI). In addition, the development upgrade process results in a merged design repository that enables customers to either continue to use it as their new Siebel development environment in the OCI tenancy, or export, and import to their on-premise instance to continue development activities there.

Key benefits include:

- Perform Siebel CRM Upgrades in a cost efficient, low risk manner with a shorter timeframe
- Dramatically reduce lengthy provisioning cycles
- Easy to repeat, and run the development upgrades multiple times using the DevOps pipeline
- Flexibility to move your application with business customizations intact to Oracle Cloud Infrastructure (OCI), if needed by business
- Monitor and evaluate Upgrade issues rapidly

Supported Languages and Versions of Siebel CRM

Upgrade Factory enables the efficient transition from Siebel CRM 8.0 and above to the latest Siebel CRM release update. The 22 supported languages are—Arabic, Simplified Chinese, Traditional Chinese, Czech, Danish, Dutch, English, Finnish, French, German, Hebrew, Italian, Japanese, Korean, Polish, European Portuguese, Brazilian Portuguese, Russian, Spanish, Swedish, Thai, and Turkish.

Prerequisites

These are the few prerequisites to keep in mind before you begin installing the upgrade factory:

- Minimum supported application version—Siebel CRM version 8.0 and above

- License to Siebel CRM Base Application
- Oracle Cloud Infrastructure (OCI) Tenancy
- Appropriate permissions to create quotas for each compartment
- Compute capacity: 12 Oracle CPU (OCPU), 32GB RAM allocated under OCI
- System requirements: Oracle Linux 7 and above
- Storage space requirements: 200GB

Before you start the upgrade, you must export the database dump using the Oracle Data Pump utility **expdp** (applies to Oracle Database).

Note: Use the `repimexp` command to export a repository.

Connect as the system database administrator or **sysdba** and run the following command.

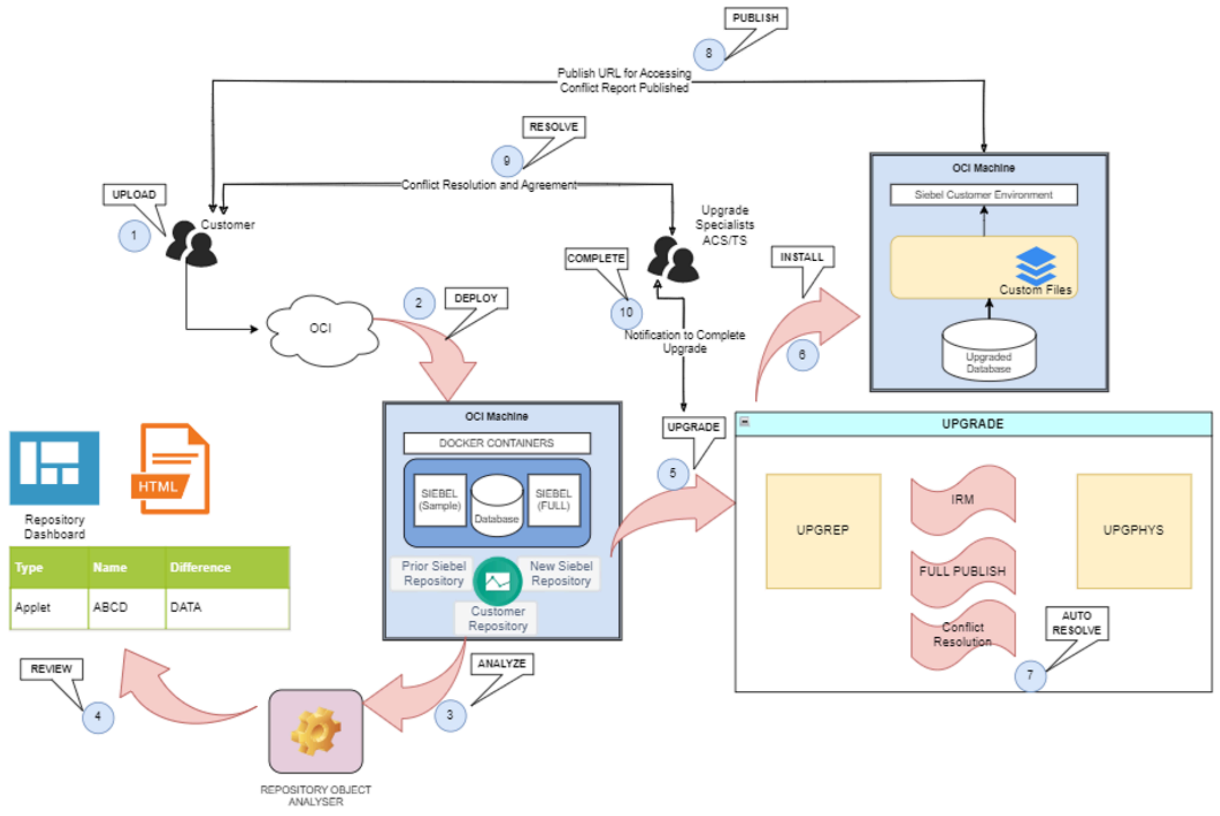
```
CREATE OR REPLACE DIRECTORY BACKUP_DIR AS <DirectoryPathOnDBServer>;
grant create any directory to <TableOwner>;

GRANT READ, WRITE ON DIRECTORY BACKUP_DIR TO <TableOwner>;
expdp <TableOwner>/<TableOwnerPassword> directory=BACKUP_DIR dumpfile=<DumpFileName>nologfile=y parallel=8
 compression=all exclude=statistics
reuse_dumpfiles=y
```

The Upgrade Process Flow

Here's the upgrade process flow in a sequence:

1. Upload the database or repository custom files to OCI
2. Begin the upgrade through the SCM REST API endpoint
3. Generate the pre-upgrade assessment report to analyze risks, if any
4. Review the assessment report
5. Create an upgrade event
6. Install the Siebel software to run the upgrade
7. Prepare the conflict report
8. Publish the URL to access the conflict report
9. Resolve conflicts on the web tools
10. Complete development upgrade and take the repository back on premise (if required)
11. Access link from SCM

The following figure explains the upgrade process flow.



Type	Name	Difference
Applet	ABCD	DATA

Pre-upgrade Assessment

You can conduct a pre-upgrade assessment using the customer repository, which includes an in-depth review of the existing customizations, infrastructure, business processes and providing repository conflicts summary and complexity scores. With the pre-upgrade assessment, you can assess how many conflicts can be expected by object type; what are the most critical conflicts you might encounter, and so on.

The pre-upgrade report identifies critical configuration issues identified in the repository, repository merge conflicts and recommendations to ensure you have a successful upgrade and limited downtime.

Before you run the pre-upgrade assessment tool, you need the following:

- Database dump or repository
- Complete extract of the following folders:
  - SWT (`<Siebel_root>/webtempl`)
  - Public (`<Siebel_Root>/objects/<lang>/srf`)
  - CFG (`<Web_Server_Root>/public`)
- Environment architecture diagram
- Completed pre-assessment questionnaire

The pre-upgrade assessment does the following:

- Assesses the custom configurations
- Identifies the extensibility of the application
- Provides details of integration in Siebel CRM
- Identifies potential conflict areas
- Applies filters for usage pattern tracking and application-level details
- Identifies complex modules and customizations

API Endpoints

You can perform operations with different REST APIs. Here are the available APIs:

- `PUT /scm/api/v1.0/upgradefactory` — Rerun upgrade environment
- `DELETE /scm/api/v1.0/upgradefactory` — Cleanup upgrade environment
- `GET /scm/api/v1.0/upgradefactory` — Get upgrade status
- `POST /scm/api/v1.0/upgradefactory` — Trigger upgrade with payload

Setting up and Configuring SCM with Upgrade Factory

Set up and configure Siebel Cloud Manager (SCM) with the upgrade factory feature.

To Create the SCM Stack

To create an SCM stack you need to identify the compartment for it and generate private and public keys.

To identify a compartment for the SCM stack

1. Log on to the OCI Console - *https://www.oracle.com/in/cloud/sign-in.html*.
2. Enter your tenancy name, that is, your cloud account name.
3. Sign in with your Oracle Cloud credentials.
4. Select your preferred region (for example, Canada Southeast)

   Note: It works on any region but if there are resource limits on any region, the stack creation might fail.

5. To select an OCI compartment, search for compartments.
6. Create a new compartment (for example, uf-scm-test) by clicking **Create Compartment**.

   Note: Skip this step if you want to use an existing compartment.

7. Copy the compartment OCID and keep it handy in a notepad.

To generate API keys for the OCI user (PEM key)

1. For the API key fingerprint, click the profile icon and drill down to your username.
2. Navigate to the API keys on the left pane of the Detail page.
3. If you don't have an API key, click **Add API Key**.
4. Download the private and public PEM key file and keep it safe.
5. Click **Add**.
6. Copy the configuration file preview information and click **Close**.

To generate the SSH private key

1. Go to any directory on your system and launch the command interface.
2. Run the command `ssh-keygen` and enter an SSH key file name at the prompt (Windows - cmd , MacOS - Terminal).

   ```
   ssh-keygen
   Enter file in which to save the key (/Users/prashanth/.ssh/id_rsa): <filename> Ex: id_rsa_scm
   Enter passphrase (empty for no passphrase): <Leave it blank and hit enter>
   Enter same passphrase again: <Leave it blank and hit enter>
   ```

Your identification is saved in **id_rsa_scm**.

Your public key information is saved in **id_rsa_scm.pub**.

The private and public key information is stored in your current directory. Keep this information safe.

Create the SCM Upgrade Factory Instance

To create the upgrade factory instance

1. In the OCI console, go to **Market Place** > **All applications**.
2. Search for SCM.
3. Click **Siebel Cloud Manager (SCM) with Upgrade Factory**.
4. Select a compartment of your choice. Keep your OCID handy.
5. Accept the conditions and click **Launch stack**.
6. Update the name of your choice (optional) and leave the other default options untouched.
7. Enter the compartment OCID that you kept handy.
8. Using the `key-gen` command browse the SSH public key information.

9. Enter a resource prefix name of your choice.

   > **Note:** This is the value which you enter for the field **Resource Prefix to name the OCI resources**, and keep it handy to identify the compartment of your SCM VM later (for example, **UpgradeFactory**)

10. Select **use User Principal** under permissions.
11. Enter the fingerprint details that you kept handy in your OCID.
12. Upload the OCI PEM private key which was downloaded for the same fingerprint.
13. No passphrase is required.
14. Select the CloudManager instance configuration.
15. Select CloudManager Instance Type: VM Standard E4 Flex.
16. Enter the **OCPU** value as 8 and **Memory** as 32 GB.
17. Select **Assign Public IP Address**.
18. Click **Next** and review the details on the summary page.
19. Select the **Run Apply** checkbox and click **Create**.
20. Wait for a couple of minutes while the instance is created. Keep monitoring the logs and wait for the state to change to **Succeeded**.
21. Scroll down to the end of log and copy the SCM instance details available.
22. Copy the PUBLIC_IP and instance URL and save it in your notepad. This PUBLIC_IP refers to the OCI VM instance created for SCM and upgrade factory execution.

    > **Note:** Keep this PUBLIC IP address handy for all your future transactions.

23. To access the SCM application, Open the URL and try launching the SCM URL which you copied from your notepad.
24. Click and expand **v1.0 environments-related operations** to find the list of APIs available (Upgrade Factory API details available at the end).

Preparing the Upgrade

Prepare to Execute Upgrade Factory

Upgrade Factory REST API Endpoint URI List

Prepare your list of REST API End point Uniform Resource Identifiers (URIs) by replacing <PUBLIC_IP> with the IP address which is assigned to your SCM OCI VM Instance:

- **POST** - http://<PUBLIC_IP>:<Port>/scm/api/v1.0/upgradefactory - To trigger upgrade process with payload
- **GET** - http://<PUBLIC_IP>:<Port>/scm/api/v1.0/upgradefactory - To get upgrade process status anytime
- **DELETE** - http://<PUBLIC_IP>:<Port>/scm/api/v1.0/upgradefactory - To cleanup upgrade env before triggering the fresh request
- **PUT** - http://<PUBLIC_IP>:<Port>/scm/api/v1.0/upgradefactory - To retrigger the upgrade from the failed point

Upload Files on OCI Object Storage Bucket (OSB)

To upload files to the bucket

1. Log on to Oracle Cloud.

2. Navigate to **buckets** and drill down on the **Buckets Object Storage**.
3. Pick your preferred compartment and region.
4. If you want to create a new bucket, click **Create Bucket**. Leave the default options untouched and click **Create**.
5. Drill down to the bucket name.
6. Click **Upload** and browse the dump or repository file, which you want to upload .
7. Click **Upload** at the bottom and wait for the upload to complete.

Create a Pre-Authenticated Request URL

To create a pre-authenticated request URL

1. Go to **More options** > **Create pre authenticated request**.
2. Select an expiration date of your choice or leave it default (1 week).
3. Click **Create pre authenticated request**.
4. Copy the pre-authenticated request URL and keep it safe.

> **Note:** This pre-authenticated URL can be used to download files from the OSB until the request expires.

Set up Boomerang for REST API Calls

Boomerang is a client application used to trigger REST API calls. However, you can use any REST API client of your preference to trigger the API requests.

To set up Boomerang

1. Search for **Boomerang SOAP REST** Chrome extension on the Chrome browser.
2. Install the Boomerang Chrome extension.
3. Click the Chrome Extensions (next to the address bar on the chrome browser) or under Chrome > Options, and launch Boomerang.
4. In the Boomerang application, click **New Request** on the top.
5. Enter the API URI in the address bar of the application and select the input body format from the dropdown as **JSON**.
6. Enter the input payload data on the left pane and select the type of request (POST/GET/DELETE/....) you want to trigger from the dropdown.
7. To trigger the request, click **Send**. You will see the response in the right pane.

> **Note:** REST API endpoints are protected and you need to enter a username and password. Enter these credentials:
> - USERNAME - admin
> - PASSWORD - get a token from the SCM instance by signing in with an SSH key. Go to the folder `/home/opc/config` and view the `api_creds.ini` file.
>
> Make a note of the `basic_auth_password`.

Executing the Upgrade

Prepare the Payload

Supported Siebel Version List

Accepted Siebel Input Version (current_siebel_version)	Corresponding Siebel Version
v8.0	8
v8.1.1.10sia	8.1.1.10
v8.1.1.11sia	13
v8.1.1.14sia	14
v8.2.2sia	8.2.2
v8.2.2.3sia	8.2.3
v8.2.2.4sia	13
v8.2.2.14sia	14
v15.0	15
v15.5	15.5
v16.0	16

List of Siebel Component Groups

Component groups	Names
adm	Application Deployment Manager
asgnmgmt	Assignment Management
commmgmt	Communications Management
contctr	Content Center
dataqual	Data Quality
mobilesync	Disconnected Mobile Synchronization
mobilesyncsis	Disconnected Mobile Synchronization SIA
dandb	Dun and Bradstreet
eai	Enterprise Application Integration
fieldsvc	Field Service
fcstsvc	Forecast Service Management
sync	Handheld Synchronization
syncsis	Handheld Synchronization SIA
icomp	Incentive Compensation

mktgom	Marketing Object Manager
mktgsrv	Marketing Server
rtsremote	MWC Real Time Sync
pimsi	PIM Server Integration Management
creditasgn	Sales Credit Assignment
saleshiersvc	Sales Hierarchy Service
search	Search Processing
siebanywhere	Siebel Anywhere
callcenter	Siebel Call Center
communications	Siebel CME
cra	Siebel Core Reference Application
eautomotive	Siebel eAutomotive
echannel	Siebel eChannel
econsumer	Siebel eConsumerSector
edocuments	Siebel eDocuments
hospitality	Siebel eHospitality
erm	Siebel Employee Relationship Management
fins	Siebel Financial Services
htim	Siebel High Tech Industrial Manufacturing
sisme	Siebel Industry Marketing
iss	Siebel ISS
lifesciences	Siebel Life Sciences
loyalty	Siebel Loyalty
loyaltyengine	Siebel Loyalty Engine
publicsector	Siebel Public Sector
remote	Siebel Remote
rti	Siebel RTI
sales	Siebel Sales
ucm	Siebel Universal Customer Master
siebelwebtools	Siebel Web Tools
taskui	Task UI
workflow	Workflow Management

xmlpreport	XMLP Report

Input Payload Details

Parameter	Value Mandatory	Purpose	Description
requester_email	Yes	Email Id of the requestor	Array of user email id
current_siebel_version	Yes	Siebel version of the Customer Repository	Accepted Siebel version
component_groups	No	Component groups to be enabled on the upgrade application	Comma separated values
base_language	Yes	Mandatory parameter. Base language of the Customer Siebel Application Supported languages : 22 languages	Single language entry (Example: enu)
other_languages	No	Currently not supported	This is a required entry and should be left blank till notified on the support
upt_file_url	No	OOS Bucket PAR URL for zipped file containing all UPT csv file (<filename>.zip)	Files to be zipped directly inside the zip file without any enclosing directory (zip <filename>.zip file1 file2 file3 ....)
webtmpl_file_url	No	OOS Bucket PAR URL for the zipped file containing custom webtemplates (<filename>.zip)	Files to be zipped directly inside the zip file without any enclosing directory (zip <filename>.zip file1 file2 file3 ....)
js_file_url	No	OOS Bucket PAR URL for the zipped file containing custom js files (<filename>.zip)	Files to be zipped directly inside the zip file without any enclosing directory (zip <filename>.zip file1 file2 file3 ....)
css_file_url	No	OOS Bucket PAR URL for the zipped file containing custom css files (<filename>.zip)	Files to be zipped directly inside the zip file without any enclosing directory (zip <filename>.zip file1 file2 file3 ....)
keyfilebin_url	No	OOS Bucket PAR URL for the zipped file containing custom keyfilebin files (<filename>.zip)	Files to be zipped directly inside the zip file without any enclosing directory (zip <filename>.zip file1 file2 file3 ....)
dump_file_url	Yes for Database Dump	OOS Bucket PAR URL for the Database Dump File (no zipping)	dump_file_url or repo_file_url is mandatory (zipped/tar files not allowed)
schema_owner	Yes for Database Upgrade	DB Schema owner for the database dump	Source database schema owner
tablespace	Yes for Database Upgrade	Table space name for the database dump	Source database tablespace

index_tablespace	No	Index tablespace name for the database dump	Source database index tablespace
repo_file_url	Yes for Repository	OOS Bucket PAR URL for the Customer Repository dat file (no zipping)	dump_file_url or repo_file_url is mandatory (zipped/tar files not allowed)

Payload Format and Sample Input Payload

Input Payload Format

```
{
 "current_siebel_version": "",
 "requester_email": ["user1@example.com","user2@example.com","user3@example.com"],
 "base_language":"",
 "other_languages":"",
 "component_groups":"",
 "upt_file_url":"",
 "custom_files":{
 "webtmpl_file_url":"",
 "js_file_url":"",
 "css_file_url":"",
 "keyfilebin_url":""
 },
 "db_dump": {
 "dump_file_url" : "",
 "schema_owner":"",
 "tablespace":"",
 "index_tablespace":""
 },
 "customer_repository": {
 "repo_file_url" : ""
 }
}
```

Sample Input Payload —Repository Upgrade

```
{
 "current_siebel_version": "v8.1.1.14sia",
 "requester_email": ["user1@example.com","user2@example.com","user3@example.com"],
 "base_language":"enu",
 "other_languages":"",
 "component_groups":"fins",
 "upt_file_url":"https://objectstorage.XXXXXXXXXXXXXX/UpgradeFactory/o/UPTCSV.zip",
 "custom_files":{
 "webtmpl_file_url":"https://objectstorage.XXXXXXXXXXXXXX/UpgradeFactory/o/WT.zip",
 "js_file_url":"https://objectstorage.ap-mumbai-1.oraclecloud.com/p/XXXXXXXXX/o/JS.zip",
 "css_file_url":"https://objectstorage.ap-mumbai-1.oraclecloud.com/p/XXXXXXXXXX/o/CSS.zip",
 "keyfilebin_url":""
 },
 "db_dump": {
 "dump_file_url" : "",
 "schema_owner":"",
 "tablespace":"",
 "index_tablespace":""
 },
 "customer_repository": {
 "repo_file_url" : "https://objectstorage.us-ashburn-1.oraclecloud.com/p/XXXXXXXXXXXX/o/
Repository_81114.dat"

 }
}
```

Sample Input Payload —Database Upgrade

```
{
 "current_siebel_version": "v8.1.1.14sia",
 "requester_email": ["user1@example.com","user2@example.com","user3@example.com"],
 "base_language":"enu",
 "other_languages":"",
 "component_groups":"fins",
 "upt_file_url":"https://objectstorage.XXXXXXXXXXXXXX/UpgradeFactory/o/UPTCSV.zip",
 "custom_files":{
 "webtmpl_file_url":"https://objectstorage.XXXXXXXXXXXXXX/UpgradeFactory/o/WT.zip",
 "js_file_url":"https://objectstorage.ap-mumbai-1.oraclecloud.com/p/XXXXXXXXX/o/JS.zip",
 "css_file_url":"https://objectstorage.ap-mumbai-1.oraclecloud.com/p/XXXXXXXXXX/o/CSS.zip",
 "keyfilebin_url":""
 },
 "db_dump": {
 "dump_file_url" : "https://objectstorage.us-ashburn-1.oraclecloud.com/p/XXXXXXXXXXXX/o/
SiebelDatabase.dmpdp",
 "schema_owner":"SIEBEL",
 "tablespace":"DATA",
 "index_tablespace":"DATA"
 },
 "customer_repository": {
 "repo_file_url" : ""
 }
}
```

To trigger the upgrade with the payload

1. Prepare the payload as per the pre-defined payload format.
2. Open any preferred REST API client (Boomerang/Postman).
3. Paste the payload in the input area of the application and select POST method.
4. Enter the REST API URI- http://<PUBLIC-IP>:<Port>/scm/api/v1.0/upgradefactory.
5. Post the request by clicking **Send** (in case of Boomerang).
6. You will be prompted for authentication credentials.
7. Enter these default credentials to proceed with the request;
   - USERNAME - admin
   - PASSWORD - get a token from the SCM instance by signing in with an SSH key. Go to the folder `/home/opc/config` and view the `api_creds.ini` file.

   Make a note of the `basic_auth_password`.

   Default Credentials

   USERNAME - admin

   PASSWORD - Token from the previous step.
8. Wait for the response, which contains all the steps you need as part of the upgrade.
9. You will receive an email for subscription confirmation from Oracle Cloud Infrastructure Notifications Service.
10. Click the **Confirm subscription** hyperlink to enable the email notifications for upgrade execution.

The following table lists the steps that are executed as part of the Upgrade Factory process:

Step	Details	Action
01_env_provisioning	Environment provisioning and Siebel setup using the Repository/dump file provided.	Automated
02_assessment_report_generation	Pre-upgrade Repository Analyzer execution and Assessment Report generation.	Automated

03_upgrep_irm_process	Trigger Siebel Upgrade Process - UPGREP followed by IRM Process.	Automated
04_launch_webtool_for_conflict_resolution	Prepares the environment to enable Webtool launch for conflict resolution.	Automated
05_conflict_resolution	Waits for user to launch webtool and complete the conflict resolution.	Launch Webtool and complete conflict resolution
06_upgphys_process	Resumes Upgrade process with full publish followed by UPGPHYS process to complete the Siebel Upgrade.	Automated

Status Check and Actions

Checking the Execution Status

After the upgrade factory process starts, it takes a couple of hours to set up the environment and execute the repository analyzer. You receive an email on completion of every step, if you subscribed to notifications. If you want to know the status of the execution, you can trigger a GET request on the same URI to know the current status (REST API URI Format: http://<PUBLIC_IP>:<Port>/scm/api/v1.0/upgradefactory).

Here's what you see:

- The Status is marked as completed on successful execution with a start time and end time.
- The current running status is marked as in-progress with a start time.
- Important URLs are listed at the bottom of the payload.
- If the upgrade status shows **failed**, it means there is an issue with the execution.

Sample Output Payload

```
{
 "data": {
 "stages": {
 "01_env_provisioning": {
 "comment": "",
 "end_time": "Mon, 05 Jun 2023 07:28:41 +0000",
 "start_time": "Mon, 05 Jun 2023 06:55:32 +0000",
 "status": "completed"
 },
 "02_assessment_report_generation": {
 "comment": "",
 "end_time": "Mon, 05 Jun 2023 09:50:31 +0000",
 "start_time": "Mon, 05 Jun 2023 07:28:52 +0000",
 "status": "completed"
 },
 "03_upgrep_irm_process": {
 "comment": "",
 "end_time": "Mon, 05 Jun 2023 13:44:30 +0000",
 "start_time": "Mon, 05 Jun 2023 09:50:37 +0000",
 "status": "completed"
 },
 "04_launch_webtool_for_conflict_resolution": {
 "comment": "",
 "end_time": "Mon, 05 Jun 2023 13:54:31 +0000",
 "start_time": "Mon, 05 Jun 2023 13:45:14 +0000",
 "status": "completed"
 },
 "05_conflict_resolution": {
 "comment": "Please launch Siebel Webtools and complete the conflict resolution to proceed further.",
 "end_time": "Mon, 05 Jun 2023 15:53:37 +0000",
 "start_time": "Mon, 05 Jun 2023 13:54:35 +0000",
 "status": "completed"
 },
 "06_upgphys_process": {
 "comment": "Upgrade is successfully completed. Please launch the application URLs and validate.",
 "end_time": "Mon, 12 Jun 2023 10:15:15 +0000",
 "start_time": "Mon, 12 Jun 2023 10:01:12 +0000",
 "status": "completed"
 }
 },
 "upgrade_status": "completed",
 "urls": {
 "check_status_url": http://144.22.242.19:16690/scm/api/v1.0/upgradefactory,
 "final_outputs": https://objectstorage.sa-saopaulo-1.oraclecloud.com/
p/3CmuPVeQV2568wYHNWgeOmitDP9154LIgBuDqnNr_MHfPblgUq7wvBoi9KQWI15v/n/siebeldev/b/UpgradeFactoryLogs-uf-scm-
beta-r2-siebel-cm/o/FinalOutputFiles230605072746.zip,
 "repo_analyzer_assessment_report": https://objectstorage.sa-saopaulo-1.oraclecloud.com/p/
t3nre_D2b3butSBedY--7eX47MAKnBrwGYxZB3zwHVsj9x2XxXAjcGIfvjVndL3y/n/siebeldev/b/UpgradeFactoryLogs-uf-scm-
beta-r2-siebel-cm/o/RepositoryAnalyzerReport230605072746.zip,
 "siebel_fins_enu": https://144.22.242.19/siebel/app/fins/enu,
 "siebel_smc_url": https://144.22.242.19/siebel/smc,
 "siebel_webtool_url": https://144.22.242.19/siebel/app/webtools/enu,
 "upgrade_factory_logs": https://objectstorage.sa-saopaulo-1.oraclecloud.com/p/
Xz2QWcyNQnv_nPHxZkJFoDbeSTBRTwU2qID9Q7lpAdB4yhHTD2zBznYy30972Ez4/n/siebeldev/b/UpgradeFactoryLogs-uf-scm-
beta-r2-siebel-cm/o/UpgradeFactoryLogs230605072746.zip
 }
 },
 "message": "Upgrade is already running.",
 "status": "success"
}
```

Pre-upgrade Assessment Report (RepAnalyzer Reports)

Here's what you see in the pre-upgrade assessment report:

- Post assessment, the UPGREP and IRM is kicked off
- The assessment report and summary dashboard are uploaded to OCI bucket and the pre-assessment Report URL is shared in the status check response.

Here's what you can do next:

1. Download the report from any browser.
2. Unzip the `ReportData.txt` and RepAnalyzer Dashboard from the downloaded file.
3. Open the `summary.html` file inside the RepAnalyzer Dashboard directory.
4. The summary shows a count of conflicts that are critical, needs attention and informational for UI, business, and data entities, in the form of this table:

Classification	Definition
Info	No action required from the customer
Attention	Customer has to verify and resolve, but not a show stopper
Critical	Need action from customer. It's can be a show stopper

5. Click **Conflict Count** to navigate to the detailed page.
6. The tree structure on the left provides the categorization of UI, Business and Data layer.
7. Only critical conflicts are displayed in the table.

8. You can filter by critical, needs attention and informational by using the checkboxes.
9. You can filter conflicts analyzed based on UPT data.

Add Ingress Rules on OCI to Enable Access to Server

All the connections to the host server ports are secured using Ingress rules configuration on Oracle cloud infrastructure (OCI). If you are unable to access any of the required ports like 443 for the Siebel application and 1521 port for the database connect, it might be due to missing ingress configuration.

Here's how you can configure ingress rules:

1. Identify your sub compartment name under the root compartment which you selected when you created the SCM stack.

   Compartment name format : `<resource_prefix_name> -siebel-cm`

   `resource_prefix_name` is the resource prefix to name the OCI resources. For example, if `resource_prefix_name = UpgradeFactory` then the compartment name is `UpgradeFactory-siebel-cm`.
2. Go to **OCI console** > **Compute** > **Instances** and select the Compartment ID (resource prefix name + -siebel-cm) and click **Instance Name**.
3. Drill down to the **Name** hyperlink for the instance having your Public IP, and then drill down further on the **Subnet** hyperlink.
4. Select **security-list-for-cm**.
5. Add ingress rules for required ports:
   - Source CIDR - this is the IP address of the source machine which would need access to the server. To get the IPv4 address of the machine, use this link: *https://whatismyipaddress.com/*. Source CIDR is <youripaddress>/32
   - Destination port range - 443

   Note: Port 443 for Siebel application access including webtool, 1521 port for Siebel DB access.
6. Click **Add Ingress Rules**.

Resolve Conflicts using Webtools

The Upgrade conflict resolution screen is now enabled on Siebel webtools. Once the upgrade factory process completes the **04_launch_webtool_for_conflict_resolution** step, Status check response provides you a webtools URL, which is also sent to you by email. At this point, the upgrade factory process is paused for user action. Make sure the ingress rule is added for your system IPAddress and port 443. (Webtool URL is hosted with port 443).

Here's how you can resolve conflicts using Webtools:

1. Launch Webtools using the URL available in the payload response.
2. Enter the credentials shared on Webtools.
3. Navigate to **Tools** > **Upgrade conflicts**.
4. You can choose to override using the override flag or select options using the **Resolution** dropdown.
5. Select multiple records on the screen and, or, click **Change records** to update the records.
6. Click **Accept recommendations** to update the recommendations on all records.
7. Click **Conflict Resolution Complete** on the top applet and accept the confirmation message.
8. On Confirmation, Siebel webtools is logged out and the upgrade factory process will resume in sometime.

Complete the Upgrade

Here's what happens when the upgrade factory process resumes:

- Once the conflict resolution is marked as completed, UPGREP is triggered.

- Upgrade is complete once the UPGPHYS is completed.
- The application URLs are available at the bottom of the response.
- You can access the final output files using the URLs in the response.
- The zipped file contains the Repository export and schema ddl.
- Execution is now full completed and the upgraded application is available.

Troubleshooting Upgrade Process Issues

Accessing the Database Dump URL

During the pre-assessment, if you get an error while accessing the Database dump URL

- Check connectivity and see if the database is down.
- To resolve the issue, make sure the database dump or repository URL you specified is correct.

Accessing Webtools

When you try to access the Webtools after clicking **Conflict Resolution complete**, if you get an error message

- The server you are trying to access is either busy or experiencing difficulties.
- Close the web browser, open a new browser window, and try logging in again.

Note: Webtools will logoff and will not be accessible till further steps of upgrade are completed.

Upgrade Factory Execution Failure

If the Upgrade Factory execution failed due to an unexpected error

- There are several likely reasons for this error, and you should check your alert logs for details.
- Use this link to browse the log directory and view the error log files: `http://<PUBLIC_IP>:<Port>/scm/api/v1.0/upgradefactory/logs`

Frequently Asked Questions

How to view log files?

Any failure in the execution is captured in the upgrade factory logs.

To view log files

1. Log on to the Siebel application container.
2. Use the API link below to get the log files URL for logs download. This is a zipped file with all the logs related to Upgrade Factory execution.
3. Trigger a GET request using the REST API Client: `http://<PUBLIC_IP>:<Port>/scm/api/v1.0/upgradefactory/logs`

How to connect to the host server?

Mac users can use the Launch terminal application. Windows users can install a terminal application such as gitbash (*https://git-scm.com/downloads*)

To connect to the host server

1. Place the SSH key in any of the folders (This is the same private key which was used while creating the SCM stack)
2. Open the terminal window and navigate to the folder having the SSH key. For example: `cd/user/johndoe/demo/ocikey)http://<PUBLIC_IP>:<Port>/scm/api/v1.0/upgradefactory/logs`
3. Change the permission to 600.

   ```
   chmod 600 <SSH Private Key Name>
   ```
4. Establish connection using the command below.

   ```
   ssh -i <SSH Private Key Name> opc@<PUBLIC_IP>
   ```

   Example: `ssh -i scmuf-private.key opc@111.111.111.111`
5. If the connection is not established, then a Wrong SSH private key is being used or the company firewall is blocking the connection.
6. Disconnect the VPN and try again.

How to connect to Siebel server?

Once you are logged into the VM Instance <PUBLIC_IP> using an SSH Terminal, you can log on to different Siebel servers using the Docker commands.

Login to Siebel container.

```
docker exec -it <container name> bash

example (to connect to enterprise server): docker exec -it ses-ENT bash
```

How to copy a file or folder inside a Docker container to the host VM instance?

Use this command:

```
docker cp <container name>:<file/folder inside the container with full path> <host server path> example(to
copy /siebel/ses/siebsrvr/test.txt file inside enterprise server to /home/opc): docker cp ses-ENT:/siebel/ses/
siebsrvr/test.txt /home/opc/
```

Use this command from your local machine terminal to copy file from host server to your local machine.

```
scp -i<SSH Private Key Name> -r opc@<PUBLIC_IP>/<file/folder> <localmachine-path> example:scp -iscmuf-private-
key – r opc@111.111.111.111/home/opc/test.txt ~/Downloads/
```

How to connect to the database?

Connect to Database from ses-ENT container (Siebel Enterprise Server).

```
SQLPLUS Connection: sqlplus sadmin/Welcome1@SIEBELDB or sqlplus siebel/Welcome1@SIEBELDB

ODBC Connection: odbcsql /u sadmin /p Welcome1 /s SIEBELDB
```

Connect to the database using SQL Developer.

By default, Siebel instances expose the Siebel DB on port 1521 with the service name of SAMPLE.

To connect

1. Launch SQL Developer.
2. Click on the **+** icon on the left pane to add a new database connection and fill the connection details as shown below and click **Test** to verity the status (success or failure).
    - **Name**: My SCM Siebel (any name of your choice)
    - **Username**: SADMIN
    - **Password**: Welcome1
    - **Hostname**: <{PUBLIC_IP}>
    - **Port**: 1521
    - **Service name**: SAMPLE

12 Siebel CRM Upgrade Factory On Premise

Siebel CRM Upgrade Factory On Premise

This chapter provides information on the following topics:

- *Overview of the Siebel CRM Upgrade Factory On Premise*
- *Preparing for the Upgrade*
- *Installation and Deployment*
- *Triggering the Upgrade*
- *Executing the Upgrade*
- *End to End Process Steps*

Overview of the Siebel CRM Upgrade Factory On Premise

Upgrade Factory On Premise enables a simpler, faster, more predictable upgrade process. Once the Siebel CRM application is upgraded, customers can take advantage of the latest innovations and adopt monthly updates via the continuous release model.

Key benefits include:

- Makes the transition of Siebel CRM 7.8.2 and higher to the latest release easier.
- Enables customers to easily run Upgrade.
- Speeds deployment time with quick setup of a development environment.
- Identifies conflicts and fix ups that are needed before the upgrade.
- Completes an upgrade in a few clicks with as little human intervention as possible.

Supported Database Platforms, Operating Systems and Versions of Siebel CRM

With Upgrade Factory – On Premise, you can run Upgrade in a platform-agnostic way. It supports various OS/Db combinations across many operating system (Oracle Linux 7, Solaris, Microsoft Windows, IBM AIX) and database platform (Oracle Database, Microsoft SQL, IBM DB2 UDB).

Note: IBM DB2 for z/OS is currently not supported.

Upgrade Factory On Premise enables the efficient transition from Siebel CRM 7.8.2 and above to the latest Siebel CRM release update.

Prerequisites

These are the few prerequisites to keep in mind before you begin installing the upgrade factory on premise:

- Minimum supported application version Siebel CRM version 7.8.2 and above.
- License to Siebel CRM BaseApplication.
- System requirements: Any Operating System and Database Platform supported by Siebel CRM as documented on My Oracle Support.

Note: IBM DB2 for z/OS is currently not supported.

Preliminary Setup

Before you start the Upgrade, you must complete these prerequisites:

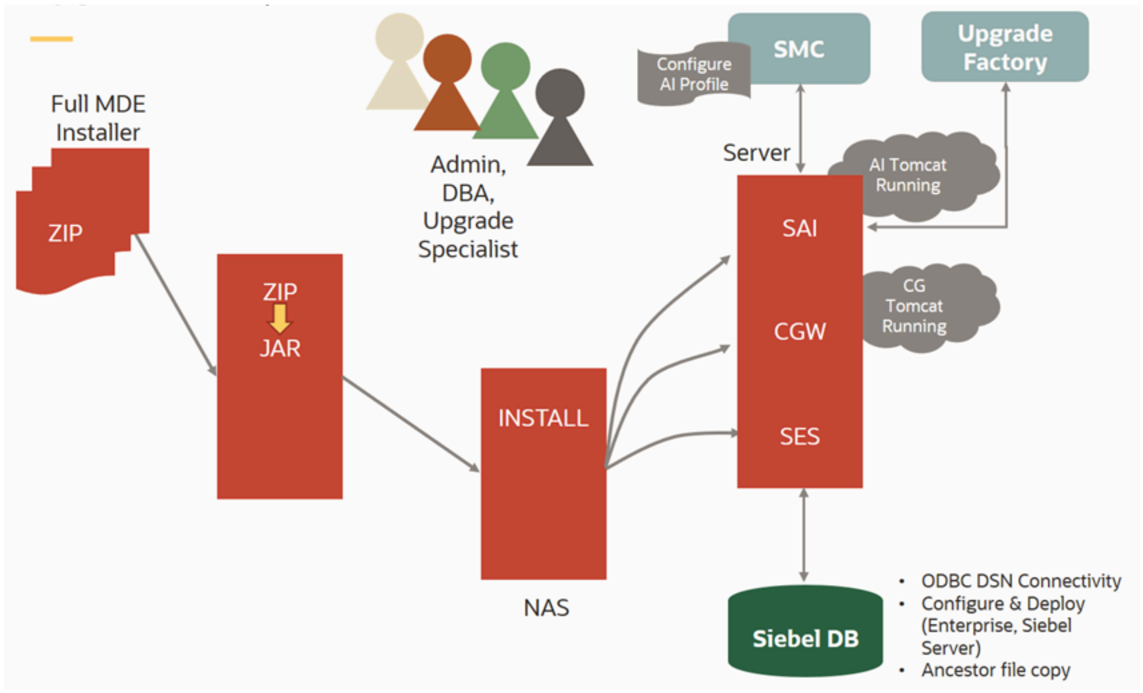
- Install Siebel on the server (Windows/Linux/AIX/Solaris).
- Single server installation (install Enterprise, Siebel server, Application Interface in a single server environment).
- Ancestor repository file corresponding to the existing Siebel version to be downloaded, and placed on server (dbsrvr/common/sia folder).
- Execution server should have connectivity to database server and Open Database Connectivity (ODBC) connection must be established before the kickoff.
- All files provided in the input payload should be accessible on the server box.

The Upgrade Process Flow

Here's the upgrade process flow in a sequence:

1. Prepare the customer DB server and Upgrade server for Siebel setup.
2. Install Siebel using installation and setup wizard.
3. Deploy Siebel Gateway and Enterprise using SMC.
4. Preparation
5. Upgrade Factory Trigger via REST API.
6. Execute UpgRep
7. Launch Web Tools for Conflict Resolution.
8. Complete Upgrade (UpgPhys).
9. Launch the upgrade applications to verify the database upgrade completion.

The following figure explains the upgrade process flow.



1. Source: MOS, eDelivery
2. Download, Unzip, Install, Base and Patch.
3. AI and CG Tomcat are running
4. SMC is used to Bootstrap
5. Configure Gateway, Enterprise Server and temporary deployment, and deletion of Siebel server .

   Note: This prepares the server with all the required environment variables and server setting.

6. Upgrade Factory is a REST Resource configured in AI.
7. Bring down all the Siebel Services (applicationcontainer_internal, gateway and applicationcontainer_external), check the ODBC connectivity on the server and, then bring up the applicationcontainer_external.
8. Trigger Upgrade Factory REST API POST Request with Payload.
9. Monitor the progress using Upgrade Factory REST API GET Request.
10. DB Backup after merge.
11. Start Siebel Services, deploy Siebel server with web tools and other required AOMs enabled.
12. Deploy Application Interface Profile with Web tools and other AOMs enabled.
13. Launch Web Tools to review and resolve conflicts.
14. Resume the Process using REST API PUT request after conflict resolution to proceed with Full Publish and UpgPhys completion.
15. Launch the upgrade applications to verify the database upgrade completion.

API Endpoints

You can perform operations with different REST APIs. Here are the available APIs:

POST: To post an Upgrade request with required input payload

```
https://<ServerIP or host>:<port>/siebel/v1.0/upgradefactory/upgrade
```

GET: To get the Upgrade execution status at any point in time

```
https://<ServerIP or host>:<port>/siebel/v1.0/upgradefactory/upgrade
```

PUT: To re-trigger the process from any failed point

```
https://<ServerIP or host>:<port>/siebel/v1.0/upgradefactory/upgrade
```

DELETE: To clear the instance

```
https://<ServerIP or host>:<port>/siebel/v1.0/upgradefactory/upgrade
```

Preparing for the Upgrade

Preparation of customer DB server and Upgrade server for Siebel setup

- Prepare a DB server having customer DB with old version of Siebel.
- Identify a Server box to install Siebel.
- Download the installation package on to the server box.
- Download the ancestor repository corresponding to customer version of Siebel.

Installation and Deployment

Siebel installation using installation and setup wizard

1. Generate Java Keystore file.
2. Initiate the installation wizard.
3. Install Siebel enterprise components and application container components along with database utilities.
4. Follow the installation wizard and pick the respective ports for Siebel containers and provide a valid Java KeyStore.

Siebel gateway and Enterprise deployment using SMC

1. Login to SMC console.
2. Register the gateway with required ports.
3. Deploy Siebel enterprise.
4. Deploy Siebel server (temporary).
5. On Successful deployment of Siebel server, delete the Siebel Server deployment for time being.

Upgrade Factory deployment

1. Prepare the environment by creating all the required directories for the input.
2. Place the custom files, web templates, ancestor repository file etc. in their respective directories on the server.

Triggering the Upgrade

Upgrade Factory Trigger via REST API

1. Prepare the input payload data with all the relevant information.
2. Install required API Client like POSTMAN or Boomerang which is required to Trigger the API requests (on your local machine).
3. Trigger a POST request with input payload to start the Upgrade Factory execution.

Executing the Upgrade

Development Upgrade

UpgRep + IRM Execution

1. Siebel database upgrade process is started.
2. UpgRep: Database schema is upgraded.
3. Take Database backup, gather stats and resume the process to start repository merge.
4. IRM: Repositories are merged and all conflicts are recorded for conflict resolution (Full merge in case of V8.0).

Web Tools launch for Conflict Resolution

1. Siebel server profile is re-deployed.
2. Web Tools component is started, and Web Tools URL (AI Profile) is deployed.
3. Launch Web Tools application on any browser and override conflicts wherever required and see recommendations from Upgrade Factory Smart Analyzers.
4. Click on conflict resolution complete button to confirm the completion.
5. Resume the process to trigger FullPublish and UpgPhys process.

UpgPhys (Upgrade Completion)

1. FullPublish is triggered post completion of conflict resolution.
2. UpgPhys: Triggered post completion of FullPublish.
3. Upgrades the physical schema and performs certain updates and cleanup on the database to prepare the final upgraded database.
4. Application will be running with upgraded database.

Sample Input JSON Payload

Trigger Upgrade (API-POST) with Payload

Note: Some of the parameters highlighted may not be relevant for your Database. Not application parameters can be blank or can be removed from the request rest all the inputs are mandatory.

```
{
 "execution_preferences": {
 "execution_type": "Dev"
 },
 "siebel_details": {
 "current_siebel_version": "v16.0",
 "base_language": "enu",
 "other_languages": ""
 },
 "siebel_server_details": {
 "siebel_home": "/scratch/home/sblusr/ses",
 "server_host": "siebel.company.com",
 "ai_port": "9001"
 },
 "database_details": {
 "database_platform": "ORACLE",
 "tblo": "SIEBEL",
 "tblo_user": "SIEBEL",
 "tblo_password": "********",
 "admin_user": "SADMIN",
 "admin_password": "********",
 "tablespace": "DATA",
 "index_tablespace": "DATA",
 "tablespace16k": "",
 "tablespace32k": "",
 "odbc_datasource": "siebel_DSN",
 "tnsname": "siebel"
 },
 "files": {
 "ancestor_repo_file": "/scratch/home/sblusr/fs/upgfiles/r160.dat",
 "custom_directory": "/scratch/home/sblusr/fs/custfiles",
 "webtemplate_directory": "/scratch/home/sblusr/fs/webtmpl"
 }
}
```

To trigger the upgrade with the payload

1. Prepare the Payload as per the pre-defined payload format.
2. Paste the payload on the Input area of the application (Boomerang/Postman) and POST the request by clicking on the SEND button (in case of Boomerang).
3. Wait for the response.
4. Response will contain all the steps which would be executed as part of the Upgrade.

Response JSON Payload

```
{
 "upgrade_status": "Complete",
 "comments": "Execution complete",
 "data": {
 "01_environment_setup": {
 "status": "Complete",
 "comment": "",
 "start_time": "11-07-2024 13:05:42 GMT",
 "end_time": "11-07-2024 13:06:12 GMT"
 },
 "02_upgrep_process": {
 "status": "Complete",
 "comment": "",
 "start_time": "11-07-2024 13:06:12 GMT",
 "end_time": "11-07-2024 14:43:49 GMT"
 },
 "03_database_backup": {
 "status": "Complete",
 "comment": "",
 "start_time": "11-07-2024 14:43:49 GMT",
 "end_time": "11-07-2024 16:26:16 GMT"
 },
 "04_irm_process": {
 "status": "Complete",
 "comment": "",
 "start_time": "11-07-2024 16:28:50 GMT",
 "end_time": "12-07-2024 03:03:46 GMT"
 },
 "05_webtool_preparation": {
 "status": "Complete",
 "comment": "Please wait while we prepare the environment for webtool deployment.",
 "start_time": "12-07-2024 03:03:46 GMT",
 "end_time": ""
 },
 "06_conflict_resolution": {
 "status": "Complete",
 "comment": "",
 "start_time": "12-07-2024 00:24:40 GMT",
 "end_time": "12-07-2024 06:24:40 GMT"
 },
 "07_fullpublish": {
 "status": "Complete",
 "comment": "",
 "start_time": "12-07-2024 06:26:09 GMT",
 "end_time": "12-07-2024 09:49:15 GMT"
 },
 "08_upgphys_process": {
 "status": "Complete",
 "comment": "",
 "start_time": "12-07-2024 09:49:15 GMT",
 "end_time": "12-07-2024 16:28:56 GMT"
 }
 },
 "references": {
 "upgrade_factory_logs": "...ses/siebsrvr/log/UpgradeFactory",
 "upgrade_logs": "...ses/siebsrvr/log"
 }
}
```

Status Check and Actions

Checking the Execution Status

To get upgrade status

1. Trigger a REST API - GET request with no input payload.
2. Wait for the Response.
3. Response payload will have status **completed** for all the steps which are complete and **InProgress** status for the step which is currently in progress.

Here is what you see in the response payload:

- Element `upgrade_status` of `InProgress` indicates that the Upgrade Factory is running without any issues.
- Element **comments** provided current step, status or action related information.
- Important URLs are shared under **references** element in the response payload.
- If `upgrade_status` is failed, then the execution has encountered an issue and has failed.

End to End Process Steps

Here is the end to end steps from start to finish:

1. Installation
   - Initiate the installation using setup executable.
   - Pick the home location for Siebel installation.
   - Pick relevant languages.
   - Select Installation Components, Siebel Enterprise Components, New Database Repository (master repository) and Configuration Tasks Enterprise Container Configuration, Application Interface Container Configuration .
   - Assign unused ports for application containers.
   - Set a temporary credentials for SMC Login (Example: admin/admin).
   - Provide KeyStore and TrustStore jks file and KeyStore password.
   - Save the response file and Click on install.
2. Deployment
   - Keep the required server info ready.
   - Prepare SMC URL.
   - Register Gateway host and port.
   - Update TLS port (Example: 9111).
   - Update Database details for Gateway security profile.
   - Enter the Database credentials for ADMIN user (SADMIN).
   - Re-login to SMC using Database credentials.
   - Setup Gateway registry.
   - Create Enterprise Profile.
   - Enter the TNS name for Oracle.
   - Deploy Enterprise Profile.
   - Wait for deployment status to turn green in the deployment tree.
   - Create Siebel Server Profile.
   - Select Siebel Enterprise Application Integration, Siebel Web Tools.
   - Select the default application of your choice.
   - Deploy Siebel Server.
   - Pick the relevant languages.
   - Wait for Siebel Sever deployment to turn green.
   - Once the deployment is successful, Force Delete the deployment.

3. Preparation

    - Open a fresh OS session (New terminal in case on Non-windows).
    - Shutdown `Applicationcontainer_internal` (Non-windows: Use command).
    - Shutdown Gateway Services (Non-windows: Use command).
    - Shutdown Applicationcontainer_external (Non-windows: Use command).
    - Startup Applicationcontainer_internal (Non-windows: Use command).

4. Execution

    - Prepare required payload data for Upgrade Factory Trigger.
    - Download and place the relevant ancestor repository on the server.
    - Open any REST API client (Example: Boomerang chrome extension).
    - Create JSON payload with all required information.
    - Prepare Upgrade Factory REST API endpoint URI.
    - Verify if Siebel application container external is up and running.
    - Provide basic Database authentication details for the API Requests.
    - Trigger POST request with required input JSON payload.
    - Trigger GET request to know the current status of execution.
    - Once the RepAnalyzer execution is complete, launch Summary URL from the response.
    - Conflict Summary is available on the home screen.
    - Drilldown on the infolet to get the critical conflict details.
    - Check the filter buttons to view attention and info records.
    - Table shows only the Critical conflicts.
    - Process starts UpgRep execution.
    - Upon completion of UpgRep, process is paused for Database backup.
    - Process can be resumed once Database backup is done and STATS gathered.
    - Trigger a PUT request to resume the process after backup.
    - Trigger GET request to know the latest status of execution.
    - After Merge completion, Database is prepared to launch Web Tools.
    - Process is paused to complete Conflict resolution using Web Tools.
    - Launch SMC portal for enabling Web Tools.
    - Deploy the Siebel Server Profile that was deleted during preparation.
    - Select relevant languages.
    - Wait for deployment status to turn green.
    - Create Application Interface Profile.
    - Fill in all the required details.
    - Add Applications for Web Tools Object Manager.
    - Deploy Application Interface Profile.
    - Launch Web Tools using the Web Tools URL.
    - Navigate to Upgrade Conflicts screen.
    - Change the Resolution value using Override slider button.
    - Update multiple records using Change Records button.
    - Once the conflicts are resolved, Click on Conflicts Resolution Complete.
    - Web Tools Application is logged out.
    - If required, take a Database backup and Trigger a PUT request to resume the process.

- Trigger GET request to know the latest status of execution.
- After FullPublish completion, final Upgrade process UpgPhys is kicked off.
- Upgrade is complete with the completion of UpgPhys.
- Launch the upgraded applications to verify the database upgrade completion.

13 Reviewing the Siebel Upgrade Log Files

Reviewing the Siebel Upgrade Log Files

This chapter provides information on accessing, reviewing and archiving Siebel upgrade log files. This chapter includes the following topics:

- *Summarizing Siebel Log Files Using the Logparse Utility*
- *Reviewing Siebel Upgrade Log Files for Errors*
- *Manually Archiving Siebel Upgrade Log Files*

Summarizing Siebel Log Files Using the Logparse Utility

Environments: Development, production test, production.

Platforms: Windows and UNIX only. Does not apply to IBM z/OS.

Use the Logparse utility to analyze and summarize the log files created when you run the Upgrade Wizard.

The Logparse utility writes its findings to a summary file:

- Windows: `SIEBEL_ROOT\log\process\summary.html`
- UNIX: `$SIEBEL_ROOT/log/process/summary.html`

In these paths, process is the upgrade process you want to review, for example, `upgrep_dev_8x`.

The Logparse utility also writes a summary.txt file that contains the same information as summary.html. Use the summary.txt file if you do not have a browser.

After writing the output files, the Logparse utility opens summary.html automatically.

Command-Line Syntax for the Logparse Utility

The following table lists the command-line syntax for the Logparse utility.

`Flag`	Parameter	Description	Comment
`/s`	install_dir	Full path to the parent directory of SIEBEL_ROOT ($SIEBEL_ROOT). For example, if SIEBEL_ROOT is `C:\sba8x\siebsrvr`, then install_dir is `C:\sba8x`. Enclose the path in quotes.	Required
`/g`	LANGUAGE_CODE	The language in which the Upgrade Wizard ran. This is called the resource language, and typically is the language in which you ran the Siebel Installation	Required

Flag	Parameter	Description	Comment
		Wizard. For example, if the resource language is English, then the language code is enu.	
`/r`	PROCESS	Name of the schema process for which you want to generate a summary file, for example, **`upgrep_dev_8x`**. To obtain the schema process name, look in **`siebsrvr\log`**. The subdirectory names in **`log`** are the schema process names.	Required.
`/l`	LOGPARSE_FILENAME	Name of the Logparse log file.	Default value is **`logparse.log`**
`/n`	MAX_NUMBER_SQL	Maximum number of longest-running SQL statements to display in the summary.	Default value is 10
`/t`	THRESHOLD_TIME	Threshold time for longest running SQL statement, in the format hh:mm:ss.	Default value is 00:10:00 (By default, SQL statements that run longer than 10 minutes are displayed)
`/e`	MAX_NUMBER_ERRORS	Maximum number of errors to display in the summary.	Default is 8

Running the Logparse Utility

Use the following procedure to run the Logparse utility.

To run the Logparse utility

1. Navigate to the following directory:

   Windows: **`SIEBEL_ROOT\bin`**

   UNIX: **`$SIEBEL_ROOT/bin`**
2. Run the following command. Use the syntax in the following table:

```
logparse /s install_dir /g LANGUAGE_CODE /r PROCESS /l LOGPARSE_FILENAME /n MAX_NUMBER_SQL /
t THRESHOLD_TIME /e MAX_NUMBER_ERRORS
```

   Windows example (Upgrade Wizard language is English):

```
logparse /s C:\sea8xx /g enu /r upgrep_dev_8xx /l logparse.log /n 10 /t 00:00:10
/e 10
```

3. Review the Logparse log and verify that Logparse ran without errors.

   Windows: `SIEBEL_ROOT\bin\logparse.log`

   UNIX: `$SIEBEL_ROOT/bin/logparse.log`

Related Topics

About the Siebel Database Upgrade Log Files

Reviewing Siebel Upgrade Log Files for Errors

Reviewing Siebel Upgrade Log Files for Errors

Environments: Development, production test, production.

Platforms: Windows and UNIX only. Does not apply to IBM z/OS.

Using the Logparse utility to review the log files created when you run Upgrade Wizard is the recommended way to verify that the upgrade process completed correctly and to identify errors that must be resolved.

The Logparse utility writes its findings to a summary file:

Windows: `SIEBEL_ROOT\ log\ process\summary.html`

UNIX: `$SIEBEL_ROOT/log/process/summary.html`

In these paths, process is the upgrade process you want to review, for example `upgrep_dev_8x`.

Logparse also writes a summary.txt file that contains the same information as summary.html. Use the summary.txt file if you do not have a browser.

The procedures in this topic describe how to review the summary.html file.

Verifying the Upgrade Environment

Before reviewing the upgrade steps for errors, verify that the upgrade environment is set up correctly.

To verify the upgrade environment

1. If summary.html is not already open in your browser, then navigate to `siebsrvr\log`, and locate the subdirectory for the upgrade process you want to review.
2. In the desired upgrade process directory, click summary.html to open the file in a browser.
3. On the summary.html home page, Upgrade/Install Log File Navigation Page, locate the Parameter column in the table, and click the Parameters link.

   The Parameters page displays.
4. Review the parameters, and verify that they are correct.

   License key values are encrypted. If you find errors in the parameters, then correct them, and run the upgrade process again as needed.

Determining Whether the Upgrade Process Completed Successfully

The upgrade processes consist of a series of steps. Each step must complete successfully.

If the Upgrade Wizard cannot complete a step, then it marks the step incomplete in the state log (`siebsrvr\log\Process\state\state.log`) and exits. You must correct the error, and run the Upgrade Wizard again. When you rerun the Upgrade Wizard, it refers to the state log and resumes at the incomplete step that contained the error.

Use the following process to identify errors:

1. Resolve errors for steps with a Status of Incomplete.
2. Review all the steps with a Status of Complete. If any contain unacceptable errors, then resolve these errors.
3. Restart the Upgrade Wizard, or if necessary, restore the database and rerun the upgrade process.

If you have any questions regarding how to resolve errors, then contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.

To determine whether the upgrade process completed successfully

1. If summary.html is not already open in your browser, then navigate to `siebsrvr\log`, and locate the subdirectory for the upgrade process you want to review.
2. In the desired upgrade process directory, click summary.html to open the file in a browser.
3. On the summary.html home page, Upgrade/Install Log File Navigation Page, locate the Step/Error column in the table and click the Steps/Errors link.

   The Steps/Errors Summary page displays.
4. Locate the Status column, and verify that all the steps have a status of Complete.

   If the Status of all the steps is Complete, then the upgrade process completed successfully.

   If the Status of any step is Incomplete, then the upgrade process did not complete successfully. The Upgrade Wizard has exited before completing all the steps in the upgrade process. You must identify the error, and correct it.

   > **CAUTION:** In some cases, the Upgrade Wizard can complete a step even though the step contains unacceptable errors. You must verify that all steps do not contain unacceptable errors.

Determining Whether Steps Contain Unacceptable Errors

There are two types of unacceptable errors:

- The Upgrade Wizard could not complete a step due to an error and exited. You can locate these errors by checking the Status field on the Steps/Errors Summary page. If the Status of a step is Incomplete, then the step contains an unacceptable error.
- The Upgrade Wizard encountered an unacceptable error but was able to complete the step. You can locate these errors by examining the detailed information for each step.

To determine whether steps contain unacceptable errors

1. If summary.html is not already open in your browser, then navigate to `siebsrvr\log`, and locate the subdirectory for the upgrade process you want to review.
2. In the desired upgrade process directory, click summary.html to open the file in a browser.
3. On the summary.html home page, Upgrade/Install Log File Navigation Page, locate the Step/Error column in the table, and click the Steps/Errors link.

   The Steps/Errors Summary page displays.
4. In the Name column, click the name link.

   The detailed information for the step displays. An alternate method is to use the slide bar to browse down through the detailed information listing for all the steps.
5. IBM DB2 and Oracle Database: In the detailed information for each step, check the Errors item.

   If the value for Errors is None, then the step has no unacceptable errors.

   If table lists one or more errors, then these errors are unacceptable and must be corrected. Click the links to the log files to obtain more information about the errors.
6. Microsoft SQL Server: In the detailed information for each step, check the Errors item.

   If the value for Errors is None, then the step has no errors.

   If there is a table or a link to a table listing errors, then do the following:

   a. Open the errors.xls spreadsheet:

      Windows: `DBSRVR_ROOT\DATABASE_PLATFORM\errors.xls`

      UNIX: `DBSRVR_ROOT/DATABASE_PLATFORM/errors.xls`

      If an errors.rtf (Windows) or errors.txt (UNIX) file is present, then you can use either of these files. They contain the same content as errors.xls. The errors file lists benign errors that do not need to be corrected.
   b. Compare the errors listed in the Logparse summary.html file to the list of acceptable errors documented in the error file.

      The log files generated by Upgrade Wizard appear in the errors file as `upgwiz1.log`, `upgwiz2.log`, incrementing for additional log files.
   c. If you find the error in the errors file, then the error is acceptable and no action is required.

      If you find an error that is not listed in the errors file, then the error is unacceptable. You must correct the condition that caused the error.

If you cannot resolve an unacceptable error, then contact Oracle Global Customer Support. *Do not proceed with the upgrade*.

Reviewing Upgrade Performance

The Logparse summary file provides the input that Upgrade Tuner uses to analyze the performance of the upgrade scripts. For example, the summary file provides Upgrade Tuner with information about SQL commands that returned 0 rows.

If you plan to use the Upgrade Tuner to tune your production upgrade, then familiarize yourself with the performance information in the summary file.

To review upgrade performance in the summary file

1. If summary.html is not already open in your browser, then navigate to `siebsrvr\log`, and locate the subdirectory for the upgrade process you want to review.
2. In the desired upgrade process directory, click summary.html to open the file in a browser.
3. On the summary.html home page, Upgrade/Install Log File Navigation Page, locate the following columns:
    - Performance Information for SQL
    - Performance Information for DDL
4. Click the links in these columns to display performance information about the upgrade process.

    The Performance Information for SQL page displays the longest running queries and queries that returned 0 rows for SQL scripts.

    The Performance Information for DDL page displays the tables created, altered, and deleted by the ddlimp utility and by SQL scripts.

    In both pages, the Net Cost column lists the time required to perform each operation.

Related Topics

About the Siebel Database Upgrade Log Files

Summarizing Siebel Log Files Using the Logparse Utility

Manually Archiving Siebel Upgrade Log Files

Manually Archiving Siebel Upgrade Log Files

Environments: Development, production test, production.

Platforms: Windows and UNIX only. Does not apply to IBM z/OS.

After a successful installation and upgrade, you must manually save and archive the log files located in the `SIEBEL_ROOT\log\PROCESS` (Windows) directory.

By default, only nine upgrade log files are retained for subsequent retries of the Upgrade Wizard. After nine log files have been created, when the Upgrade Wizard is rerun, it overwrites log files beginning with the earliest one created and recycles the rest as necessary. (This does not apply to the state.log file.)

The number of log files retained can be increased by resetting the `siebel_log_archive` environment variable to 20. For example, to retain twenty (20) log files.

14 Performing the Siebel Repository Merge

Performing the Siebel Repository Merge

This chapter provides guidelines for performing a Siebel Repository merge. This chapter includes the following topics:

Preparing for the Repository Merge

Environments: Development environment only.

The process of merging repositories to create a final customized repository used in the upgrade is time-intensive and resource-intensive. As a result, a merge might fail due to environmental factors, such as space constraints. When this happens, the merge process continues, even if there is a fatal database error, and the errors might go undetected.

If the merge fails, then you must restore the database environment to its premerge state and run the repository merge again. Additionally, it is recommended, as another precaution, that you export the New Customer Repository to preserve any existing workflows.

Recovering from a Failed Merge

It is recommended that you perform one of the following two tasks to preserve the premerge environment in the event of a failed merge:

- **Perform full database backup prior to the merge.** Back up the entire database prior to the merge. If the merge fails, then you can restore the database to its premerge state, and rerun the merge operation.
- **Export the New Customer Repository prior to the merge.** Export the New Customer Repository prior to the merge to create a backup copy. If the merge fails, then delete the failed repository, then import the backed up copy of the New Customer Repository. See *Using Siebel Tools* for information on exporting and importing repositories using the Database Configuration Wizard.

Note: If you export the New Customer Repository, then you must truncate the following merge log tables: S_MERGE_LOG, S_MERGE_LOG_OBJ, and S_MERGE_LOG_ATTR.

Configuring Siebel Repository Objects to Inherit Upgrade Behavior

Environments: Development environment only.

You can link objects together so that one object inherits the upgrade behavior of another. You do this by specifying an upgrade ancestor for an object.

You can specify an upgrade ancestor for the following object types:

- Applet
- Business component
- Integration object
- Report

Before doing the repository merge, review new objects you have created and determine whether you want to specify an upgrade ancestor.

Specifying an Upgrade Ancestor

Use the following procedure to specify an upgrade ancestor.

To specify an upgrade ancestor

1. Navigate to the object in Siebel Tools.
2. Click in the Upgrade Ancestor field.

   A dialog box appears. It lists available upgrade ancestors.
3. Select the desired upgrade ancestor and click Pick.

Viewing the Descendants or Copies of an Object

Use the following procedure to view the descendants or copies of an object.

To view the descendants or copies of an object

1. Right-click the object.
2. Select View descendants from the picklist.

   A dialog box appears and lists the descendants.

Related Topics

About the Siebel Repository Merge

About Inheriting Upgrade Behavior in a Siebel Upgrade

Performing a Siebel Repository Merge

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

This task merges customizations in the Prior Customer Repository (your current repository) into the New Customer Repository (the repository in the new Siebel release).

The repository merge typically takes five to seven hours to complete.

Requirement: The workstation on which the merge will occur must have a minimum of 512 MB of RAM and at least 2 GB of virtual memory or a 2–GB page file. Inadequate system resources is one of the most common causes of repository merge failures. If your repository includes a large amount of customization, then additional memory might be required.

> **Note:** Before running a repository merge, verify that `EnableSafeboot=true` is present in the `[InfraObjMgr]` section of the tools.cfg file.

To merge the repository

1. Verify that all Siebel Tools projects are checked in and unlocked.
2. Close network connections to all repository users and exit Siebel Tools.
3. Open the tools.cfg file in the new Siebel release. It is located in the following directory:

   ```
   $SIEBEL_HOME\bin\lang_code
   ```

   In this path, `lang_code` is the language, for example enu.
4. Locate the SIEBEL section, and verify that the parameters are set as shown in the following example.

Parameter	Value
`EnableToolsConstrain`	`FALSE`
`SymStrPrefix`	`X_`

5. Save the file and close it.
6. Navigate to Control Panel, System, Advanced, Performance Settings, and then Visual Effects.
7. Click Adjust for best performance.
8. Start Siebel Tools in the new Siebel release, using the following command:

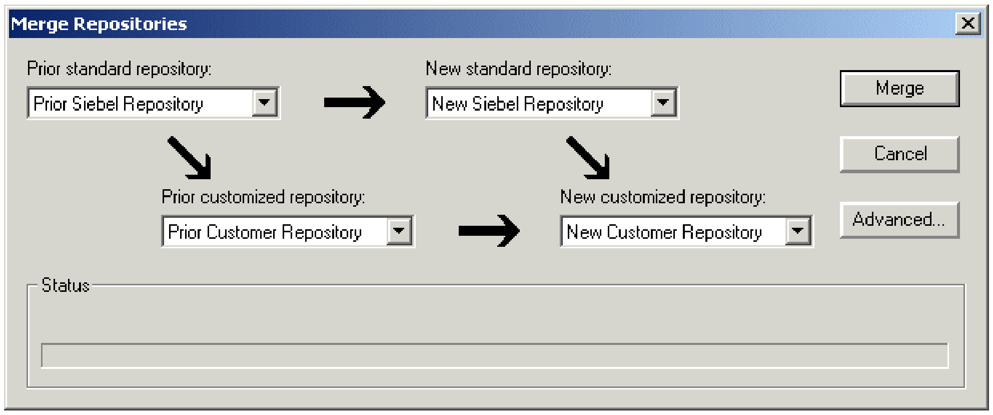
   ```
   $SIEBEL_HOME\bin\siebdev
   ```

Note: The merge0.txt file will be generated in the `$SIEBEL_HOME\bin` directory.

9. From the Tools menu, choose View, Options, and then Language Settings.
10. Verify that the language mode setting is set as desired.

    This will be the user interface language for Siebel Runtime Repositories based on the New Customer Repository. It will also be the language used by the postmerge utilities.
11. From the File menu choose the Open Repository command to open the Prior Customer Repository.
12. From the Tools menu, choose Upgrade, then the Upgrade Application menu item.

    The Merge Repositories dialog box appears.



Merge Repositories

Prior standard repository: Prior Siebel Repository

New standard repository: New Siebel Repository

Prior customized repository: Prior Customer Repository

New customized repository: New Customer Repository

    The Merge Repositories dialog box provides four options:

    - **Merge.** This button merges the repositories you specify to produce a New Customer Repository.
    - **Cancel.** This button cancels the repository merge and exits the Merge Repositories dialog box.
    - **Advanced.** This button opens the Merge Options dialog box.
13. In the Merge Repositories dialog box, choose the repositories listed in the following table.

Drop-Down List Item	Value to Choose
Prior Standard Repository	Prior x.x Siebel Repository, as appropriate for the version from which you are upgrading
Prior Customized Repository	Prior Customer Repository
New Standard Repository	New Siebel Repository
New Customized Repository	New Customer Repository

14. Click Advanced.

    The Merge Options dialog box appears.

15. In the Merge Options dialog box, click the check boxes to activate or deactivate the merge options:

    - **Abort merge if more than x errors occur.** Activate this option to abort the repository merge automatically if more than a designated number of errors occur.

      > **CAUTION:** The typical Repository merge generates many benign errors. If you select this option, then set the number of errors to a large value. This will help prevent the Repository merge from aborting due to benign errors.

    - **Incorporate Custom Layout.** Activate this option to help preserve field and button placement on prior custom or modified forms, views, and screens. Select a prior release and style for label placement.

      After you have made your selections, click OK.

      The Upgrade Check List dialog box appears.

16. In the Upgrade Check List dialog box, verify that each requirement has been met. When they have been met, select all the check boxes and click Continue.

    > **CAUTION:** The upgrade might fail if all the items in the checklist are not completed.

    After the merge completes, a dialog box appears, requesting that you make a backup of the New Customer Repository.

Related Topics

About the Siebel Repository Merge

Reviewing the Siebel Repository Merge Log Files

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

To determine whether the repository merge was successful, review the merge log files. The merge is successful if it completes without unacceptable errors:

- **Acceptable errors.** If an ancestor object is specified in an object definition, and the ancestor object is not present in the New Siebel Repository, this causes a merge error. This is an acceptable error and can be ignored.

  Here is an example of an acceptable error in the merge log file, `merge0.txt`:

  ```
  !!ERROR:CANNOT upgrade objects which have Briefing Tracking Profile Applet -
  Product marked as 'Upgrade Anc'
  ```

- **Unacceptable errors.** All other types of merge errors are unacceptable errors and mean that the merge was not successful.

  For more detailed information, go to 477269.1 (Article ID) on My Oracle Support, This document was previously published as Siebel Troubleshooting Steps 19.

Merge errors are displayed in the Upgrade Applications Objects List view in Siebel Tools. Additional details on merge errors are located in the repository merge log:

```
$SIEBEL_HOME\bin\merge0.txt
```

Each time you run the merge process, the name of the merge0.txt file is incremented, for example to merge1.txt.

If your repository merge process terminates and is flagged as Incomplete, then navigate to the Screens menu in Siebel Tools, and choose the Application Upgrader menu item. The most common reasons for its failure are:

- The number or errors (!!ERROR) exceeds the number that was predefined in Siebel Tools when the merge was started.
- The merge process has been terminated due to a local issue on the Siebel Tools workstation, such as a scheduled reboot.
- RDBMS errors caused the process to stop.
- Memory allotment issues on the workstation on which Siebel Tools is installed.
- Network failure.

If the repository merge terminates and is flagged as Incomplete, then the merge must be restarted, as discussed in *Performing a Siebel Repository Merge* .

To determine whether the repository merge was successful

1. From the Screens menu in Siebel Tools, choose Application Upgrader, and then Application Upgrade Object List.
2. In the Application Upgrades list, select the record of the merge.
3. Review the entry in the Status column:
   - **Completed.** Indicates the merge completed without errors.
   - **Completed with Errors.** Indicates the merge contains errors.
     If the Status column indicates Completed, then no further action is required. The merge was successful.
     If the Status column indicates Completed with Errors, then you must review the errors to determine whether the merge was successful. To review the errors, complete the remaining steps in this task.

     > **Note:** These errors do not indicate an incomplete merge and rerunning the merge will not correct them.
4. In the Object Differences list, click Query.
5. In the Status field, enter the following: `!!ERROR::*`
6. Press Enter to run the query. A list of objects where the merge process encountered errors is returned.
7. Open the merge log file, merge0.txt, which is located in the following directory:

   ```
   $SIEBEL_HOME\bin\merge0.txt
   ```

   If there are multiple files, then open the one with the highest number in the file name, for example merge1.txt.
8. To locate merge errors in the file, search for the `!!ERROR` string. Informational messages are marked as `!!INFO`.
9. Use the objects displayed in the Object Differences list and the errors displayed in the log file to analyze the errors:
   - If all the errors are acceptable, then the merge is considered successful. It is advisable, however, to consider the number of acceptable errors when determining whether to re-run the merge operation.
   - If the log contains unacceptable errors, then the merge has failed.
10. If the merge contains unacceptable errors, then go to 477269.1 (Article ID) on My Oracle Support. This document was previously published as Siebel Troubleshooting Steps 19. This document explains the meaning

of many of the error messages that can appear in the log file. Use this document to correct the errors. If you cannot resolve all the errors, then contact Oracle Global Customer Support.

11. Open the workflow merge log file:

    ```
    $SIEBEL_HOME\bin\merge0_ver.txt
    ```

    If there are multiple files, then open the file with the highest number in the file name, for example merge1_ver.txt. This log file is created by the workflow premerge and postmerge steps.
12. Review the log file. If the file contains errors, then create a service request (SR) on My Oracle Support, or contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.

Related Topics

About the Siebel Repository Merge

Upgrade Planning for Siebel Workflow Designer

Generating Siebel Enterprise Integration Manager Temporary Columns

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

The repository merge process does not preserve Siebel Enterprise Integration Manager processing columns for custom mappings. You must generate the missing custom Siebel Enterprise Integration Manager processing columns again.

To generate Siebel Enterprise Integration Manager temporary columns

1. In Siebel Tools, from the File menu, choose Open Repository, and then New Customer Repository.
2. From the Tools menu, choose, Upgrade, and then Generate EIM Processing Columns.

   A dialog box displays.
3. Click OK to generate Siebel Enterprise Integration Manager processing columns for custom mappings.
4. In the Object Explorer window, choose New Customer Repository and verify that the Comment field shows `UpgEimCol`.

   This indicates that the Siebel Enterprise Integration Manager temporary columns were created successfully.

Related Topic

About the Siebel Repository Merge

Reviewing Siebel Repository Object Property Conflicts

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

You can change how object property conflicts were resolved during the repository merge.

Note: Only critical conflicts are displayed when you are resolving conflicts following the repository merge.

How Object Property Conflicts Occur

The repository merge compares repository object properties in the Prior Siebel Repository, Prior Customer Repository, and New Siebel Repository. When the value of an object property is different in all three repositories, an object property conflict exists.

This occurs when you have changed the value of an object property in the Prior Customer Repository, and the value of this property has also changed in the new release (New Siebel Repository).

An object property conflict does not occur if you changed the value of an object property in the Prior Customer Repository, and the object property value did not change in the new release. When this happens, the merge process transfers the changed value to the New Customer Repository.

The merge process resolves object property conflicts by referring to the setting of the object's Standard Win property. For about 90% of repository objects, the merge process resolves conflicts by using the object property value in the New Siebel Repository.

Do not change the setting of the Standard Win property.

Application Upgrade Attribute List View

You can review and change how object property conflicts were resolved using the Application Upgrade Attribute List view in Siebel Tools. The Attribute Differences List in the view includes the following columns:

- **Object Name.** The name of the object.
- **Attribute.** The object property name.
- **Conflict.** The merge process puts a check mark in this field if there was an object property conflict during the merge.
- **Resolution.** Displays which property value the merge process used to resolve the conflict:
  - **Standard Value.** The property value in the New Siebel Repository was used. This value is displayed in the New Standard column.
  - **Custom Value.** The property value in the Prior Customer Repository was used. This value is displayed in the Prior Customized column.

- **Override.** Put a check mark in this column to change how the conflict is resolved. Overriding the resolution changes the property value in the merged repository. If the resolution was the Standard Value, then the attribute value is changed to the Custom Value and vice versa.

  Putting a check mark in the Override column does not change the value displayed in the Resolution column. It indicates that the displayed value was manually overridden in the merged repository.
- **Prior Standard.** Displays the value of the object property in the Prior Siebel Repository.
- **Prior Customized.** Displays the value of the object property in the Prior Customer Repository. In the Resolution column, this value is called the Custom Value.
- **New Standard.** Displays the value of the object property in the New Siebel Repository. In the Resolution column, this value is called the Standard Value.

Requirement: The repository merge must have been successful. See *Reviewing the Siebel Repository Merge Log Files*.

Reviewing Object Property Conflicts

Use the following procedure to review object property conflicts.

To review object property conflicts

1. In Siebel Tools, from the Screens menu, choose Application Upgrader, and then Application Upgrade Attribute List.
2. In the Application Upgrades list, select the record of the successful merge.
3. In the Attribute Differences list, click Query.
4. In the Attribute Differences list, click in the Conflict field so that a check mark appears.
5. Press Enter to run the query.

   The query displays a list of all object properties for which there is a conflict.
6. For each record, review the entry in the Resolution field.
7. To change the resolution, click in the Override field.

   A check mark appears. This changes the value of the object property in the merged repository.

   Avoid overriding conflicts for the following object properties.
   - Left
   - Right
   - Top
   - Height
   - Width

   Visually review these properties in the upgraded application before changing them.

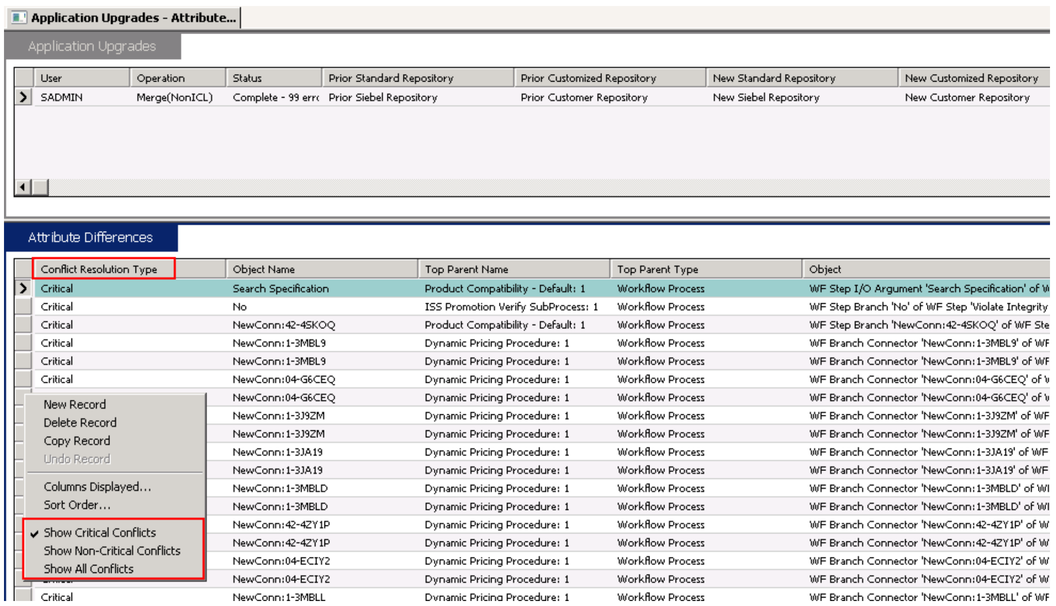
Reviewing Object Property Conflicts at the Attribute Level

Use the following procedure to review object property conflicts at the attribute level.

To review conflicts at the attribute level

1. In Siebel Tools, from the Screens menu, choose Application Upgrader, and then Application Upgrade Attribute List.

   By default, you can see only the critical conflicts in the Application Upgrades - Attribute Differences view. You must review the critical conflicts. To view informational conflicts, right-click and select Show Non-Critical Conflicts. Noncritical conflicts include conflicts of attributes, such as Sequence number, Comments, Layout, and so on.



User	Operation	Status	Prior Standard Repository	Prior Customized Repository	New Standard Repository	New Customized Repository
SADMIN	Merge(NonICL)	Complete - 99 errc	Prior Siebel Repository	Prior Customer Repository	New Siebel Repository	New Customer Repository

Conflict Resolution Type	Object Name	Top Parent Name	Top Parent Type	Object
Critical	Search Specification	Product Compatibility - Default: 1	Workflow Process	WF Step I/O Argument 'Search Specification' of W
Critical	No	ISS Promotion Verify SubProcess: 1	Workflow Process	WF Step Branch 'No' of WF Step 'Violate Integrity
Critical	NewConn:42-4SKOQ	Product Compatibility - Default: 1	Workflow Process	WF Step Branch 'NewConn:42-4SKOQ' of WF Ste
Critical	NewConn:1-3MBL9	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:1-3MBL9' of WF
Critical	NewConn:1-3MBL9	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:1-3MBL9' of WF
Critical	NewConn:04-G6CEQ	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:04-G6CEQ' of W
	NewConn:04-G6CEQ	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:04-G6CEQ' of W
	NewConn:1-3J9ZM	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:1-3J9ZM' of WF
	NewConn:1-3J9ZM	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:1-3J9ZM' of WF
	NewConn:1-3JA19	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:1-3JA19' of WF
	NewConn:1-3JA19	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:1-3JA19' of WF
	NewConn:1-3MBLD	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:1-3MBLD' of WI
	NewConn:1-3MBLD	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:1-3MBLD' of WI
	NewConn:42-4ZY1P	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:42-4ZY1P' of W
	NewConn:42-4ZY1P	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:42-4ZY1P' of W
	NewConn:04-ECIY2	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:04-ECIY2' of W
	NewConn:04-ECIY2	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:04-ECIY2' of W
Critical	NewConn:1-3MBLL	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:1-3MBLL' of WF

2. In the Application Upgrades list, select the record of the successful merge.
3. In the Attribute Differences list, click Query.
4. In the Attribute Differences list, click in the Conflict field so that a check mark appears.
5. Press Enter to run the query.

   The query displays a list of all object properties in which there is a conflict.
6. For each record, review the entry in the Resolution field.
7. To change the resolution, click in the Override field.

   A check mark appears that changes the value of the object property in the merged repository.

   Avoid overriding conflicts for the following object properties:

   - Left
   - Right
   - Top
   - Height
   - Width

   Review these properties in the upgraded application before changing them.

Related Topic

About the Siebel Repository Merge

Generating the Runtime Repository Data

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

Once the Repository merge is complete and after all merge conflicts have been resolved and before starting upgphys, you must generate the Siebel Runtime Repository data. To generate the Siebel Runtime Repository data, you must execute the Full Publish command in Siebel Tools. Full Publish must be executed against the New Customer Repository. Make sure to change the DockRepositoryName in tools.cfg to New Customer Repository and ensure Siebel tools launches against the correct Repository. If there are multiple repositories in your database, then the first preference of opening a particular repository in Siebel tools is to read from devtools.prf preference file. If the devtools.prf preference file does not exist, Siebel Tools reads from the Repository name from the DockRepositoryName parameter of the tools.cfg file. Make sure you delete the devtools.prf to ensure Siebel Tools reads the Repository name from the tools.cfg file. For more information about executing the Full Publish command in Siebel Tools, see *Using Siebel Tools* .

> **Note:** This is a mandatory step and not executing this successfully will have consequences in the further steps of an upgrade.

The Full Publish command is the following:

```
siebdev /c tools.cfg /TL ENU /d ServerDataSrc /u <username> /p <password> /
FullPublish
```

The TL `<lang_code>` parameter is a mandatory argument for Full Publish because it specifies the published languages for the database. You can add multiple languages separated by a comma. For example:

```
TL ENU,DEU,JPN
```

Depending on the number of the languages specified, the Full Publish process spans the number of Siebdev processes to compile objects in the specified language and write the data to the database.

Siebdev silently closes after completion of a Full Publish, which indicates there are no errors. If there are errors during the Full Publish process, the Siebdev process throws errors and stops.

In the log file, Failure Reported For indicates an error with an object. The log file lists the object name and type for Failure Reported For errors. Any Workflow Process errors in the log file can be ignored. In addition, the following warning errors in the log file (are acceptable and) can be ignored:

```
SBL-DAT-00144: Could not find '<?>' named '<?>'. This object is inactive or nonexistent.
SBL-DAT-00398: Field 'Extension Flag' does not exist in definition for business component ''
SBL-DAT-00388: Table 'S_RR_TABLE' defined multiple times.
SBL-DAT-60292: Root workspace is not editable.
```

Related Topic

About the Siebel Repository Merge

Regenerating the Siebel Repository Definition Files

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

If you have modified repository objects after the development environment upgrade (upgphys) and before upgrading the production test environment, then you must regenerate the schema.ddl and custrep.dat files. These files were created during the upgphys:

- **Schema.ddl.** This file contains the logical definition of the Siebel database.
- **Custrep.dat.** This file contains the definition of repository objects.

These files are used as input to the production test and production environment upgrades. These files contain only the Runtime Repository and required Design Time Repository, which is required for production environments. There is an additional custrep_dev.dat file that contains the entire development repository. This file can be created for backup purposes. If you modify the object definitions or the schema definitions in the repository after these files have been created, then you must regenerate the files.

Regenerating the schema.ddl File

Use this procedure to regenerate the schema.ddl file.

To regenerate the schema.ddl file

1. On the Siebel Server where the Siebel Database Server files are installed, navigate to the following location:

   Windows: `SIEBEL_ROOT\bin`

   UNIX: `$SIEBEL_ROOT/bin`
2. Run the following command:

   ```
   ddldict /u DatabaseOwner /p /c "ODBCDataSource" /d TableOwner
   /f DBSRVR_ROOT\DatabasePlatform\schema.ddl /e y /a y /l SiebelLogDir\sch_dict.log
   /n "Siebel Repository" /t dcir
   ```

   where:

   - DatabaseOwner is the Siebel database administrator account name.
   - Password is the Siebel database administrator account password.
   - ODBCDataSource is the ODBC name for connecting to the database. Enclose the name in quotes.
   - TableOwner is the Siebel table owner name.
   - DBSRVR_ROOT is the absolute path to the Siebel Database Server installation directory.
   - DatabasePlatform is the Siebel Database Server directory name for the database, for example Oracle. The example shows Windows path syntax. On UNIX systems, use UNIX path syntax.
   - SiebelLogdir is the path to the directory where you want the output log placed (log output directory). The example shows Windows path syntax. On UNIX systems, use UNIX path syntax.

3. After the command completes, review the output log files for errors. If the log indicates there are errors, then create a service request (SR) on My Oracle Support, or contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.

Regenerating the custrep.dat File

Use this procedure to regenerate the custrep.dat file.

To regenerate the custrep.dat file

1. On the Siebel Server where the Siebel Database Server files are installed, navigate to the following location:

   Windows: `SIEBEL_ROOT\bin`

   UNIX: `$SIEBEL_ROOT/bin`
2. Run the following command:

   ```
   repimexp /a O /u DatabaseOwner /p Password /c "$ODBCDataSource" /d TableOwner
   /r "Siebel Repository" /f $DbsrvrRoot/$DatabasePlatform/custrep.dat /l $SiebelLogDir/exprep.log
   /v Y /Y "SrcWorkspaceName=MAIN;SrcEndVer=;Lang=ALL"
   ```

   where:

   - DatabaseOwner is the Siebel database administrator account name.
   - Password is the Siebel database administrator account password.
   - ODBCDataSource is the ODBC name for connecting to the database. Enclose the name in quotes.
   - TableOwner is the Siebel table owner name.
   - DBSRVR_ROOT is the absolute path to the Siebel Database Server installation directory. The example shows Windows path syntax. On UNIX systems, use UNIX path syntax.
   - DatabasePlatform is the Siebel Database Server directory name for the database, for example Oracle.
   - SiebelLogdir is the path to the directory where you want the output log placed (log output directory). The example shows Windows path syntax. On UNIX systems, use UNIX path syntax.

     This command exports only the repository content that is required on the production environment. The entire repository is not needed on the production environment for Siebel CRM 17.0 and later releases.
3. After the command completes, review the output log files for errors. If the log indicates there are errors, then create a service request (SR) on My Oracle Support, or contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.

Related Topic

About the Siebel Database Configuration Wizard Utilities

Deleting Unneeded Siebel Repository Files

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

When you are confident that the repository has been upgraded successfully, export the New Siebel Repository and Prior Customer Repository for safekeeping. You can also delete the following repositories:

- Prior Standard Repository
- New Standard Repository

Related Topic

About the Siebel Repository Merge

Running the Siebel Postmerge Utilities

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

Note: The postmerge utilities should only be run if the previous environment was Siebel version 8.0.0.x or earlier.

After the merge is completed successfully, run the postmerge utilities. These utilities make the following changes to user interface objects in the merged repository:

- Convert certain flow-based form applets to grid-based.
- Verify that new or customized screens, views, and applets from the Prior Customer Repository are configured correctly and can be accessed in the user interface.
- Revise multi-value group (MVG) applets so that they are shuttle-enabled.
- Lists user interface objects that have missing required fields.

If the postmerge utilities encounter a problem and exit before completing, then do the following, fix the problem in the merged repository, and then rerun the utilities.

You cannot rerun the utilities if everything completes successfully, unless you restore your database to a point before they were executed.

To run the postmerge utilities

1. If you are rerunning the postmerge utilities and want to save the existing log, then rename the log. The path to the log is as follows:

   ```
   $SIEBEL_HOME\reppatch\log\reputility.log
   ```

   If you do not rename the log, it will be overwritten.
2. In Siebel Tools, from the Screens menu, choose Application Upgrader, and then Application Upgrade Object List.
3. In the Application Upgrades list, select the record of the merge.
4. In the Application Upgrades list, right-click and select Launch Post Merge Utility from the pop-up menu.

   A dialog box appears and displays the postmerge utilities log. When the utilities have finished, a message showing completion displays in the log.

Do not launch more than one instance of the postmerge utilities.

Related Topics

About the Siebel Postmerge Utilities

About the Siebel Repository Merge

15 Performing a Siebel Incremental Repository Merge

Performing a Siebel Incremental Repository Merge

This chapter provides guidelines for performing a Siebel incremental repository merge (IRM). It includes the following topics:

- *Process of Meeting Requirements for an Incremental Repository Merge*
- *Process of Performing Incremental Upgrade of Siebel Database from Siebel CRM 8.1.1.x (SIA Repository), 8.2.2.x, 15.x, and 16.x*
- *Process of Upgrading Siebel Development Environment from Siebel CRM 8.1.1.x (SIA Repository)*
- *Upgrading a Custom Database Schema*
- *About Migrating Siebel Open UI XML Manifest Data from Previous Release to New Manifest Data in the Database*
- *Default Configuration for Generic Applet Types and Configuration of Non-Applet User Interface Constituents*
- *Process of Regenerating the Siebel Repository Definition Files*
- *Performing a Production Test or Production Environment Migration from Siebel CRM 8.1.1.x (SIA Repository)*

Note: Use this chapter if you are upgrading to the latest Siebel CRM monthly update from Siebel CRM version 8.1.1.x (SIA repository only), version 8.2.2.x, version 15.x, or version 16.x. For more information about incremental repository merge as it applies to different installation options, see *Supported Upgrade Paths for Siebel CRM* and also *Siebel Installation Guide* .

Process of Meeting Requirements for an Incremental Repository Merge

Environments: Development environment only.

Before you prepare Siebel Tools for the incremental repository merge, you must perform the following tasks:

1. *Before You Begin*
2. *Preparing Siebel Tools for Custom ODBC Data Source Names on Oracle Database*
3. *Preparing Siebel Tools for Custom ODBC Data Source Names for All Databases*
4. *Editing the Siebel Tools Configuration File*
5. *Ancestor Repositories*

Before You Begin

Refer to and follow the tasks listed in *Preparing for Siebel Database Upgrade*

You must have the Siebel Enterprise Server, which includes the Siebel Gateway, Siebel Server, and Siebel Database Server (Database Configuration Utilities), installed. You must have installed the latest Siebel CRM monthly update. For installation information, see *Siebel Installation Guide* and other applicable documentation.

> **Note:** Prior to running incremental repository merge, you must have one and only one repository in your development instance and that repository must be named *Siebel Repository*. Also, if you must delete any additional repositories, the only acceptable way to do so is to select the repository record in Siebel Tools and delete it. It is not acceptable to delete repositories using direct SQL.

About the Siebel Server and Siebel Tools on Windows

It is recommended that the Siebel Server and Siebel Tools be installed on the same Microsoft Windows computer. If your Siebel Server is installed on Windows, then the merge process automatically starts Siebel Tools to execute the incremental repository merge. For this reason, the Siebel Database Configuration Wizard must be started from the computer where Siebel Tools is installed.

If the Siebel Server is not installed on the same computer as Siebel Tools, then map a network drive from the computer where Siebel Tools is installed to the computer where Siebel Server is installed. Use this mapped drive to specify the location of the Siebel Server when you run the Siebel Database Configuration Wizard.

> **Note:** Because the merge process requires extensive database resources, it is recommended that, before beginning the incremental repository merge process, you set the general cursor size to 5000.

Preparing Siebel Tools for Custom ODBC Data Source Names on Oracle Database

Platforms: Windows.

This task applies if you have a custom ODBC data source on Oracle Database. If you create a custom ODBC data source name for Oracle Database instead of using the default ODBC data source name, then you must perform the following modifications to enable Siebel Tools for the incremental repository merge. After Siebel Tools is installed, the ODBC data source named *SSD default instance* is automatically set up by the configurations, using the Oracle driver installed by the installation.

This task is a step in *Process of Meeting Requirements for an Incremental Repository Merge*.

To prepare Siebel Tools for a custom data source name on Oracle Database

1. Use the Siebel Tools default ODBC driver, Siebel Oracle90.
2. Navigate to `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\ODBC\ODBC.INI\ODBC_DSN_name`.
3. Right-click the new custom ODBC name you created, and select New, then String Value, and enter the following names:
    - ColumnsAsChar
    - ColumnSizeAsCharacter
4. After you create the values, select each value, right-click each one. For each value, choose Modify, then in the data field, enter the following values:
    - Set the ColumnsAsChar value to 1

- Set the ColumnSizeAsCharacter value to 1

> **Note:** The ODBC SSD default instance automatically has these values in the registry, but you have to manually create them for the ODBC data source that you create.

5. Click the System DSN tab of the ODBC connection that you intend to use, and then test the connection.

Preparing Siebel Tools for Custom ODBC Data Source Names for All Databases

If you create a custom ODBC data source name instead of using the default ODBC data source name, then you must perform the following modifications to enable Siebel Tools for the incremental repository merge.

This task is a step in *Process of Meeting Requirements for an Incremental Repository Merge*.

To prepare Siebel Tools for a custom ODBC data source name for all databases

1. Navigate to the `$SIEBEL_HOME\bin\<primary_lang_code>` directory, where `<primary_lang_code>` is the language code for the primary language, and open tools.cfg.
2. In the [Siebel] section of the .cfg file, do the following:
   a. Change the ServerDbODBCDataSource parameter from the default value to your custom ODBC name.
   b. (For Microsoft SQL Server and IBM DB2 only) In the [ServerDataSrc] section, change the Connect String parameter from the default value to your custom ODBC name.

Editing the Siebel Tools Configuration File

Platforms: Windows, UNIX, IBM z/OS.

This task is a step in *Process of Meeting Requirements for an Incremental Repository Merge*. Use this task to perform the required edits in the Siebel Tools configuration file.

To edit the Siebel Tools configuration file

1. Navigate to the `$SIEBEL_HOME\bin\lang_code` directory and open the tools.cfg file.
2. Make the following changes in the tools.cfg file:
   a. Change the SymStrPrefix value from the default value of X_ to SBL_.

   > **CAUTION:** You must change these values back to the default value after the merge but before modifying the Siebel Repository. It is required that you set the prefix to SBL_ to differentiate between symbolic strings you create and those provided by Siebel CRM. After the merge, to enable you to create your own custom symbolic strings, you must return the prefix back to X_. To do this, see *Editing the Siebel Tools Configuration File After the Development Environment Merge*.

   b. Make sure the EnableToolsConstrain value is set to the default value of False. If it is not, then set the value to False.
   c. Set the DockRepositoryName value from the default value of Siebel Repository to New Customer Repository.

3. Save the tools.cfg file.
4. You do this because incremental repository merge requires this prefix to be set to SBL_ so that all the symbolic strings that are included as part of the New Siebel Repository are inserted into your repository during the merge.

Process of Performing Incremental Upgrade of Siebel Database from Siebel CRM 8.1.1.x (SIA Repository), 8.2.2.x, 15.x, and 16.x

Environments: Development, production test, production.

The tasks in this process enable you to merge customizations in the Prior Customer Repository (your current repository) with the New Customer Repository (the repository in the new Siebel CRM release). This process covers upgrades from Siebel CRM version 8.1.1.x (SIA repository only), version 8.2.2.x, version 15.x, and version 16.x.

To perform incremental repository merge on your Siebel repository, perform the following tasks:

1. *Preparing to Run the Siebel Database Configuration Wizard*
2. *Requirements for Running the Siebel Database Configuration Wizard on Windows*
3. *Running the Siebel Database Configuration Wizard on Windows*
4. *Running the Siebel Database Configuration Wizard on UNIX*
5. *Preparing to Start the Siebel Upgrade Wizard*
6. *Starting the Siebel Upgrade Wizard*
7. *Stopping the Siebel Upgrade Wizard on Windows*
8. *Stopping the Siebel Upgrade Wizard on UNIX*
9. *Preparing to Restart the Merge*
10. *Restarting the Merge*
11. *Executing the RUNSTATS Command on Oracle Database During the Pause Following New Repository Creation*
12. *Executing the RUNSTATS Command on IBM DB2 During the Pause Following New Repository Creation*
13. *Executing the UPDATESTATS Command on Microsoft SQL Server During the Pause Following New Repository Creation*

Preparing to Run the Siebel Database Configuration Wizard

This task is a step in *Process of Performing Incremental Upgrade of Siebel Database from Siebel CRM 8.1.1.x (SIA Repository), 8.2.2.x, 15.x, and 16.x*. Use this task to prepare for running the Siebel Database Configuration Wizard.

Environments: Development, production test, production.

To prepare to run Database Configuration utilities

1. Identify the information that you must enter when running the Database Configuration utilities.
2. Use the following table as a guide to collect all the required information.

   After running the Database Configuration utilities and the utilities exit, you run the Siebel Upgrade Wizard. The Siebel Upgrade Wizard executes the SQL files against the Siebel database. For information on running the Siebel Upgrade Wizard, see *Preparing to Start the Siebel Upgrade Wizard*.

Field Name or Menu	Required Information
Siebel Server Directory	The absolute path of the directory where the Siebel Server is installed. For UNIX, do not enter the string `$SIEBEL_ROOT`.
Siebel Database Server Directory	The absolute path of the directory where the Siebel Database Server is installed. For example: `C:\sba81\dbsrvr`.
RDBMS Platform	Choose the RDBMS type: IBM DB2, Microsoft SQL Server, or Oracle Database. **Note:** For IBM DB2 for z/OS, see *Siebel Database Upgrade Guide for DB2 for z/OS* instead of this guide.
Siebel Database Operation menu	Choose Upgrade Database. The remaining menu choices are for database installation and administration.
Environment Type	Choose Development for development environment upgrades. Choose Production for production test environment and production environment upgrades.
Upgrade Options	Choose one of the following: • Development Environment: Upgrade Siebel Database Schema (upgrep) Upgrade Custom Database Schema (upgphys) • Production Environment: Upgrade Siebel Database Schema (upgrep and upgphys).
Siebel Industry Application	As appropriate for the environment you are upgrading from, choose SIA (for Siebel Industry Applications) or SEA (for Siebel Business Applications, also known as Siebel Cross-Industry Applications).
Current Siebel Version	Choose the repository version that you are upgrading - for more information, see *Determining Your Repository Version*. **Note:** Your repository version may not match your binary version. For example, customers on Siebel CRM 15.5 who have not run an Incremental Repository Merge from Siebel CRM 15.0 will still be on repository version 15.0.
Siebel Tools Installation Directory	The absolute path of the directory where Siebel Tools is installed, such as `C:\Siebel\8.1\Tools`. **Note:** This entry applies only to the Siebel Server installed on Windows.
Siebel Tools Data Source	Provide the data source name that you use to log in using Siebel Tools, such as ServerDataSrc. This entry is in `$SIEBEL_HOME\BIN\lang_code\tools.cfg`.
Database Encoding	Indicate whether your database uses a Unicode code page.

Field Name or Menu	Required Information
ODBC Data Source Name	Verify the ODBC name for connecting to the Siebel database that you are upgrading. If the ODBC name is not correct, then enter the correct name.
Database User Name and Database Password	Account name and password for the Siebel administrator of the Siebel database that you are upgrading. **Note:** For more information about supported characters for Siebel passwords, see *Siebel Security Guide* .
Database Table Owner and Database Table Owner Password	Account name and password for the Siebel database table owner. **Note:** For more information about supported characters for Siebel passwords, see *Siebel Security Guide* .
Index Table Space Name and Table Space Name	Oracle Database and IBM DB2 only. Index tablespace name and tablespace name (4-KB tablespace name for IBM DB2).
16-KB Table Space Name, 32K Table Space Name	IBM DB2 only. The 16-KB and 32-KB tablespace names.
Database Server OS	Choose the RDBMS server operating system type.
Parallel Indexing	Oracle Database only. Select parallel indexing if you want SQL commands for index creation to include the arguments parallel and no logging. Parallel indexing causes an index to be created using parallel processing, which requires an RDBMS server with multiple processors. Verify with your database administrator whether your RDBMS server is configured for parallel processing. **Tip:** Oracle Library search phrase: parallel execution. Selecting parallel indexing does not cause multiple indexes to be created simultaneously, in parallel. To set up parallel indexing, you must set up parallel index-creation threads, using Siebel Upgrade Tuner. You create parallel threads as part of tuning the production upgrade files. For more information, see *Tuning the Siebel Upgrade Files*
Security Group ID/Grantee	Security group or grantee name for Siebel application users. Must have select, update, insert, and delete privileges on Siebel application tables. Specify SSE_ROLE.
Custom Scripts Directory	During the upgrade process, the Repository Sanitization script is executed. This script checks the Prior Customer Repository for any unreferenced Repository objects across multiple applications. The unused objects are inactivated. If a Repository Object refers to in a custom script, it will not be a candidate for inactivation. Custom files have to be copied from the following locations: • For Siebel CRM 15.x and prior releases: `<SWSE HOME>\public\<LANG>\<BUILD#>\scripts\siebel\custom`

Field Name or Menu	Required Information
	• For Siebel CRM 16.x: `<SWSE HOME>\public\scripts\<LANG>/siebel/custom` In addition, copy every language required into a directory and specify that location here.
Web Templates Directory	Directory where the custom web templates are stored. From Siebel CRM 17.0 onwards, web templates are stored in the database. Since custom web templates must be moved to the database as well, a step is executed during the upgrade process that moves all the web templates specified in this particular location to the database. You must copy over the custom files from previous releases from: `<SES>\siebsrvr\webtempl` Copy all the custom files into a single directory and then specify that location in the upgrade path when prompted.
Verify Repository After Upgrade	Indicate whether you want to execute the steps to verify the repository during upgphys. To perform upgphys separately, select the Verify Repository After Upgrade option in the Database Configuration Wizard.
Upgrep log directory	If you select the option Verify Repository After Upgrade in the previous step, then you will have to provide the log directory of the upgrep process. The log directory is of the form: `$SIEBEL_ROOT/siebsrvr/log/upgrep_dev_UpgradeNumber` For example: `C:\ses\siebsrvr\log\upgrep_dev_811` The `log` directory path is a requirement for generating the seed data conflict report.
Log Output Directory	Specify a different subdirectory, under the `log` directory, in which to create the log files. Accept the default or enter the directory name. If the directory does not exist, then it will be created. Do not use special characters, such as spaces or slashes.
Select runupg option	Indicate whether you want to run the operation that you configured or run it at another time.

About the Database Configuration Utilities

The Database Configuration utilities are a group of wizards that request information about the upgrade process that you want to perform. The utilities add this information to a primary upgrade configuration file and call an SQL generator. The SQL generator uses the information to create or populate SQL files. For information about the required information that the utilities request when you perform an upgrade, see *Preparing to Run the Siebel Database Configuration Wizard*.

Determining Your Repository Version

Use the following procedure to determine your repository version.

To determine your repository version

1. Connect to your development database using an appropriate database-specific tool, such as Oracle's SQL*Plus.

2. Run the following query command to determine the repository version:

```
SELECT APP_VER FROM <tableowner>.S_APP_VER
```

> **Note:** Your repository version may not match your binary version. For example, customers on Siebel CRM 15.5 who have not run an Incremental Repository Merge from Siebel CRM 15.0 will still be on repository version 15.0.

After you determine your repository version, then decide which ancestor repository to use based on that version using the information in the following table. The ancestor repository relevant to your upgrade is the Siebel repository that was shipped with the Siebel CRM release that you are upgrading from. It contains none of your customizations.

> **Note:** Neither an Incremental Repository Merge nor a Database Upgrade is required for Siebel CRM 17.0 or later. For more information on installing Siebel CRM monthly updates, see the roadmap for installing Siebel CRM 22.x Update for an existing installation of Siebel CRM 17.x or later in *Siebel Installation Guide* .

Siebel Repository Version from S_APP_ VER	Upgrade Ancestor
16.0	v16_0SIA
15.5	v15_5SIA
15.0	v15_0SIA
8.2.2.14	v8_2_2_14SIA
8.2.2.4	v8_2_2_4SIA
8.2.2.3	v8_2_2_3SIA
8.2.2	v8_2_2SIA_to_v8_2_2_2SIA
8.2.1	v8_2_1SIA
8.2	v8_2SIA
8.1.1.14	v8_1_1_14SIA
8.1.1.11	v8_1_1_11SIA
8.1.1.10	v8_1_1_10SIA
8.1.1	v8_1_1SIA_to_v8_1_1_9SIA
8.0	v8_0SIA

Siebel Repository Version from S_APP_ VER	Upgrade Ancestor
7.8.2	v7_8_2

Requirements for Running the Siebel Database Configuration Wizard on Windows

This task is a step in *Process of Meeting Requirements for an Incremental Repository Merge*.

Do the following:

- Collect the information that the Siebel Database Configuration Wizard requires.

  For more information, see *Preparing to Run the Siebel Database Configuration Wizard*.
- If your Siebel Server is installed on Windows, then start the Siebel Database Configuration Wizard from the computer where Siebel Tools is installed.

  The merge process automatically starts Siebel Tools to execute the incremental repository merge.
- If the Siebel Enterprise server is not installed on the same computer as Siebel Tools, then map a network drive from the computer where Siebel Tools is installed to the computer where Siebel Enterprise Server is installed. Use this mapped drive to specify the location of the Siebel Server when you run the Siebel Database Configuration Wizard.

Running the Siebel Database Configuration Wizard on Windows

This task is a step in *Process of Performing Incremental Upgrade of Siebel Database from Siebel CRM 8.1.1.x (SIA Repository), 8.2.2.x, 15.x, and 16.x*. Use this task to run the Siebel Database Configuration Wizard on Windows.

Environments: Development, production test, production.

Platforms: Windows only.

Run the Siebel Database Configuration Wizard to upgrade the Siebel database. The Wizard collects information, populates a primary upgrade configuration file, and calls the SQL generator to create SQL commands. The Siebel Upgrade Wizard then uses the configuration file and SQL commands to execute the incremental repository merge process to incrementally upgrade the database to the newer release.

To run the Siebel Database Configuration Wizard on Windows

1. Ensure that no server tasks including the Siebel Gateway Name Service are running in the background.

   To verify this step, navigate to Start, Settings, Control Panel, and then Services.
2. Start the Siebel Database Configuration Wizard by selecting Start, Programs, Siebel SES Configuration, and then Database Server Configuration.
3. In the Siebel Database Configuration Wizard, enter the information requested in each screen, and click Next.

4. After you have entered all the requested information, the wizard displays a screen that lists the values you entered. If you want to make changes, then click Back.
5. When a dialog box is displayed asking whether you want to start the Siebel Upgrade Wizard, click Yes.
   The Upgrade Wizard starts.

Running the Siebel Database Configuration Wizard on UNIX

This task is a step in *Process of Performing Incremental Upgrade of Siebel Database from Siebel CRM 8.1.1.x (SIA Repository), 8.2.2.x, 15.x, and 16.x*. Use this task to run the Siebel Database Configuration Wizard on UNIX.

Environments: Development, production test, production.

Platforms: UNIX only.

Run the Siebel Database Configuration Wizard to upgrade the Siebel database. The Wizard collects information, populates a primary upgrade configuration file, and calls the SQL generator to create SQL commands. You then run the Siebel Upgrade Wizard to upgrade the Siebel database.

You must meet the following requirements:

- Before running the Siebel Database Configuration Wizard, you must install the language packs of the new release of Siebel CRM for each language that you deploy.
- Collect the information that the Siebel Database Configuration Wizard requires. For more information, see *Preparing to Run the Siebel Database Configuration Wizard*.

To run the Siebel Database Configuration Wizard on UNIX

1. Verify that all servers are stopped:
   a. Stop all Siebel Servers.
   b. Stop the Siebel Gateway.
2. Make `$SIEBEL_ROOT` the current directory.
3. Source the environment variables from the `siebsrvr` root directory: `install_location/siebsrvr`
   Korn shell: `siebenv.sh`
   C shell: `source siebenv.csh`
4. Review the values of the following environment variables, and confirm that the settings are correct:
   - `SIEBEL_ROOT`. This path must end in `siebsrvr`, for example, `/usr/siebel/siebsrvr`.
   - `LANGUAGE`. This is the language in which the Siebel Database Configuration Wizard runs. The value of this variable is a language identifier string. For example, `enu` is the identifier string for U.S. English.
     If either `$SIEBEL_ROOT` or `$LANGUAGE` is not set or is incorrect, then you must correct it before proceeding.
5. Start the Siebel Database Configuration Wizard:
```
$SIEBEL_ROOT/config/config.sh -mode dbsrvr
```
6. Enter the information requested in each screen, and click Next.
   After you have entered all the requested information, the wizard displays a screen that lists the values you entered.
7. If you want to make changes, then click Back.

The Siebel Database Configuration Wizard exits and prompts you to start the Siebel Upgrade Wizard (srvrupgwiz).

Preparing to Start the Siebel Upgrade Wizard

This task is a step in *Process of Performing Incremental Upgrade of Siebel Database from Siebel CRM 8.1.1.x (SIA Repository), 8.2.2.x, 15.x, and 16.x*. Use this task to run the Siebel Upgrade Wizard.

Environments: Development, production test, production.

When you run the Database Configuration Utilities on Windows, there is a prompt asking whether you want to start the Siebel Upgrade Wizard. When you run the Database Configuration Utilities on UNIX, you must start the Siebel Upgrade Wizard manually.

The Siebel Upgrade Wizard executes the upgrade of the Siebel database. The Siebel Upgrade Wizard takes a primary upgrade configuration file as input. This file contains environment information and a driver file name. The Siebel Upgrade Wizard executes the steps in the driver file to perform the upgrade.

Because the Siebel Upgrade Wizard performs the steps in the driver file, it lists the steps in a state log. The state log is located in the following directory:

```
siebsrvr/LOG/process/state
```

In the directory, process refers to the upgrade process. For example, `upgrep_dev_811sia` indicates that the upgrade is from 8.1.1, and that the upgrep process is being run in the production test or production environment.

Starting the Siebel Upgrade Wizard

This task is a step in *Process of Meeting Requirements for an Incremental Repository Merge*.

Use this procedure to start the Siebel Upgrade Wizard.

To start the Siebel Upgrade Wizard

1. Navigate to the following directory:

   - Windows: `SIEBEL_ROOT\bin`
   - UNIX: `$SIEBEL_ROOT/bin`

2. Enter the following command:

   - Windows: `siebupg /m master_UPGRADEOPTION_ENVIRONMENT_VERSION.ucf`
   - UNIX: `srvrupgwiz /m master_UPGRADEOPTION_ENVIRONMENT_VERSION.ucf`

   In these commands, `UPGRADEOPTION_ENVIRONMENT_VERSION` is the portion of the upgrade configuration file name that lists the upgrade process, the upgrade environment, and the Siebel CRM release (version) from which you are upgrading. The file is located in `SIEBEL_ROOT\bin` (UNIX: `$SIEBEL_ROOT/bin`).

   Numbers like the following examples are used for the Siebel CRM release portion of the file name:

   - 811sia
   - 81114

   The following table lists an example of the file names for an upgrade from Siebel CRM version 8.1.1 (SIA).

Upgrade Mode	File Name
Development environment upgrep	master_upgrep_dev_811sia.ucf
Development environment upgphys	master_upgphys_dev_811sia.ucf
Production environment upgrep and upgphys	master_upgrep_prod_811sia.ucf

   The following table lists an example of the file names for an upgrade from Siebel CRM version 8.1.1.11 (SIA).

Upgrade Mode	File Name
Development environment upgrep	master_upgrep_dev_81111.ucf
Development environment upgphys	master_upgphys_dev_81111.ucf
Production environment upgrep and upgphys	master_upgrep_prod_81111.ucf

3. To begin the upgrade, click OK (Windows), or press Enter (UNIX).

   The Siebel Upgrade Wizard will notify you when the upgrade process is complete.

Stopping the Siebel Upgrade Wizard on Windows

This task is a step in *Process of Meeting Requirements for an Incremental Repository Merge*.

Only stop the Siebel Upgrade Wizard if you are confident that an error has occurred and the Siebel Upgrade Wizard or a utility it has called has stopped responding. The reason for this caution is that some SQL commands issued by the Siebel Upgrade Wizard can be very time intensive and stopping and restarting the Siebel Upgrade Wizard will slow down your upgrade process.

If you are not sure whether the Siebel Upgrade Wizard has stopped responding, then create a service request (SR) on My Oracle Support, or contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.

To stop the Siebel Upgrade Wizard on Windows

Do one of the following:

- If the Siebel Upgrade Wizard has started a separate command window in which a utility is running, then close the command window. This step terminates the utility and stops the upgrade.
- In the Siebel Upgrade Wizard dialog box, click Cancel. The Siebel Upgrade Wizard exits when the current upgrade step is complete. There might be a delay while the step completes in the RDBMS.

Stopping the Siebel Upgrade Wizard can have different effects on the RDBMS. Before restarting the Siebel Upgrade Wizard, review the RDBMS log files. Run SQL commands to resolve errors found in the RDBMS log files.

Stopping the Siebel Upgrade Wizard on UNIX

This task is a step in *Process of Meeting Requirements for an Incremental Repository Merge*.

Only stop the Siebel Upgrade Wizard if you are confident that an error has occurred and the Siebel Upgrade Wizard or a utility it has called has stopped responding. The reason for this caution is that some SQL commands issued by the Siebel Upgrade Wizard can be very time intensive and stopping and restarting the Siebel Upgrade Wizard will slow down your upgrade process.

If you are not sure whether the Siebel Upgrade Wizard has stopped responding, then create a service request (SR) on My Oracle Support, or contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.

Stopping the Siebel Upgrade Wizard can have varying effects on the RDBMS. Before restarting the Siebel Upgrade Wizard, review the RDBMS log files. Run SQL commands as needed to resolve errors found in the RDBMS log files.

To stop the Siebel Upgrade Wizard on UNIX

1. If the Siebel Upgrade Wizard has started a utility in a child process, then stop the child process.
2. Exit the shell in which the Siebel Upgrade Wizard is running.
3. Locate and stop any orphaned child processes started by the Siebel Upgrade Wizard.

After the processes terminate, there might be a delay while the RDBMS executes the issued SQL commands.

Preparing to Restart the Merge

This task is a step in *Process of Performing Incremental Upgrade of Siebel Database from Siebel CRM 8.1.1.x (SIA Repository), 8.2.2.x, 15.x, and 16.x*.

Platforms: Windows, UNIX, IBM z/OS.

Before restarting the merge, consider the following requirements:

If the Siebel Upgrade Wizard encounters an error and exits during an upgrade, then you can restart it after correcting the error. The Siebel Upgrade Wizard reads the state log and continues the upgrade from the last successfully completed step.

If the Siebel Upgrade Wizard stops because of errors, then verify that you have met these requirements before restarting the wizard:

- Carefully review the relevant log files to make sure that your upgrade has completed successfully up to that point.
- Back up your complete set of log files, from the beginning of the process to the point at which it stopped, to another directory.
  This backup maintains a complete record of your log files, and it prevents your previous log files from being overwritten, which might prevent an accurate diagnosis of the reason for the error in the upgrade.
- If you are continuing a previous and incomplete schema upgrade, then do not change the path of the Log Output Directory that you previously selected.
- If problems with your environment prevent the upgrade from restarting, then you must restore the database from the prior base version; that is, the version from which you are upgrading. For example, environment problems might occur when table creation fails because of a database problem (insufficient storage or network problems), which causes subsequent upgrade steps to fail.
  If you want to restore your database and restart the upgrade, then delete or store the upgrade log files. The files are located in the following directory:
  Windows: `SIEBEL_ROOT\log\PROCESS\output`
  UNIX: `$SIEBEL_ROOT/log/PROCESS/output`
  Also delete the `state.log` file. It is located in the following directory:
  Windows: `SIEBEL_ROOT\log\PROCESS\state`
  UNIX: `$SIEBEL_ROOT/log/PROCESS/state`

Requirements for Restarting the Incremental Repository Merge

Platforms: Windows, UNIX, IBM z/OS.

If the incremental repository merge stops because of errors, then you must review the log files, and restart the merge.

It is recommended that you perform the following tasks to preserve the premerge environment in the event of a failed merge.

1. **Perform a full database backup.** Before the merge, back up the entire database. If the merge fails, then you can restore the database to its premerge state, and rerun the merge operation.

2. **Export the new customer repository.** Before the merge, export the New Customer Repository to create a backup copy. If the merge fails, then delete the failed repository, and import the backed-up copy of the New Customer Repository. See *Using Siebel Tools* for information on exporting and importing repositories using the Siebel Database Configuration Wizard.

   > **Note:** If you export the New Customer Repository, then you must truncate the following merge log tables: S_MERGE_LOG, S_MERGE_LOG_OBJ, and S_MERGE_LOG_ATTR.

Restarting the Merge

This task is a step in *Process of Meeting Requirements for an Incremental Repository Merge*.

To restart the merge

1. Restore the database to the state of the backup taken before the merge.

   You must have already executed database statistics.
2. Review the irm_sav_file.ssf file located in `$SIEBEL_HOME\bin`. The State value should equal 7 (State=7). If it is any other value, then you must change it to 7 before starting the incremental repository merge again.
3. Do one of the following:

   - **On a Windows computer.** Restart the Execute Incremental Repository Merge task by marking the step as Incomplete in the state.log file located in the Siebel Server installation directory, in `siebsrvr\log\upgrade`.
   - **On a UNIX computer.** Execute the incremental repository merge through the command line on your Windows computer.

     For more information, see *Starting the Siebel Upgrade Wizard* to restart the upgrade process.

Executing the RUNSTATS Command on Oracle Database During the Pause Following New Repository Creation

This task is a step in *Process of Meeting Requirements for an Incremental Repository Merge*. Use these tasks to run database checks on your specific database platform after your new repository has been created. The Upgrade Wizard pauses and notifies you to execute the `RUNSTATS` command and then resume. Execute `RUNSTATS` as described in the procedure that follows.

To execute the RUNSTATS command on Oracle Database

- Connect to SQL*Plus, and execute the following command:

  ```
  EXEC DBMS_STATS.gather_schema_stats (ownname => 'table_owner_name', cascade =>true,estimate_percent =>
   dbms_stats.auto_sample_size);
  ```

  Example:

  Invoke SQL*Plus from the Oracle client.

  ```
  SQL*Plus: Release 11.2.0.1.0 Production on Wed Nov 21 19:21:46 2012
  ```

```
Copyright (c) 1982, 2009, Oracle. All rights reserved.
Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining,
Oracle Database Vault and Real Application Testing options

SQL> EXEC DBMS_STATS.gather_schema_stats (ownname => 'ORAZQ108', cascade
=>true,estimate_percent => dbms_stats.auto_sample_size);
PL/SQL procedure successfully completed.
```

Executing the RUNSTATS Command on IBM DB2 During the Pause Following New Repository Creation

This task is a step in *Process of Meeting Requirements for an Incremental Repository Merge*. Use this task to execute the `RUNSTATS` command on IBM DB2, or see your database administrator to have the command executed. The Upgrade Wizard pauses and notifies you to execute the `RUNSTATS` command and then resume. Execute `RUNSTATS` as described in the procedure that follows.

To execute the RUNSTATS command on IBM DB2

- Connect to the IBM DB2 command window and execute the following commands:

```
echo "select 'runstats on table ' concat rtrim(creator) concat '.' concat rtrim(Name) concat '
 and detailed indexes all shrlevel change' concat ';' from sysibm.systables where type='T' and
 (name not in ('S_ESCL_LOG','S_DOCK_INIT_ITEM') and name not like ('S_DOCK_INITM_%')) order by
 name ;">DBname_tmp_runstats.sql
```

  Example (for UNIX operating systems):

```
db2 connect to db204205 user siebel using db2
db2 -tf DB204205_tmp_runstats.sql | grep runstats | grep -v ^select >
DB204205_runstats.sql
db2 terminate

db2 connect to db204205 user siebel using db2
db2 -tf DB204205_runstats.sql
db2 terminate
```

Executing the UPDATESTATS Command on Microsoft SQL Server During the Pause Following New Repository Creation

This task is a step in *Process of Meeting Requirements for an Incremental Repository Merge*. Use this task to execute the UPDATESTATS command on Microsoft SQL Server. The Upgrade Wizard pauses and notifies you to execute the UPDATESTATS command and then resume. Execute UPDATESTATS as described in the procedure that follows.

To execute the UPDATESTATS command on Microsoft SQL Server

1. Navigate to `$SIEBEL_HOME\bin` on the command window, and log in to odbcsql, using the following command:

```
/u SADMIN /p ******** /s ODBC_data_source_name
```

2. Execute the following command:

   ```
   Exec sp_updatestats;
   ```

Process of Upgrading Siebel Development Environment from Siebel CRM 8.1.1.x (SIA Repository)

The tasks in this process enable you to perform incremental upgrade of your Siebel development environment from Siebel CRM 8.1.1.x (SIA Repository).

Environments: Development environment only.

For more information on Siebel Database Configuration Wizard parameters, see *Preparing to Run the Siebel Database Configuration Wizard*. For information on executing the Siebel Database Configuration Wizard, see *Running the Siebel Database Configuration Wizard on Windows*.

Perform the following tasks:

1. *Upgrading the Siebel Database Schema*
2. *Executing Incremental Repository Merge on UNIX*
3. *About Siebel Repository Merge Errors*
4. *Reviewing the Siebel Repository Merge Log Files*
5. *Reviewing the Hierarchical Merge Report*
6. *Reviewing Siebel Repository Object Property Conflicts*
7. *Reviewing Log Files for All Upgrades*
8. *Reviewing Conflicts at the Attribute Level*
9. *Marking Conflict Resolution as Complete Using Siebel Tools*
10. *Generating the Runtime Repository Data*

Upgrading the Siebel Database Schema

This task is a step in *Process of Upgrading Siebel Development Environment from Siebel CRM 8.1.1.x (SIA Repository)*. Use this task to upgrade the Siebel database schema.

Note: Before you upgrade the Siebel database schema using the upgrep process, it is strongly recommended to back up the entire database. This step is mentioned in the following table (Step 24. Take backup of the Siebel database and update database statistics). If this upgrade operation fails, then you can restore the database to its preupgrade state, and rerun the upgrade operation.

To upgrade the Siebel database schema

1. Execute the Siebel Database Configuration Wizard by specifying the following parameter values:
   - Environment Type: Development
   - Upgrade Options: Upgrade Siebel Database Schema (upgrep)
2. Start the Siebel Upgrade Wizard.

The Siebel Upgrade Wizard reads the steps from the driver file, which is determined by your Siebel CRM version. The Siebel Upgrade Wizard performs the commands listed in the steps. The driver file shown in the following table contains the following information:

- Environment: Development
- Upgrade mode: upgrep
- Database: Oracle
- Multilingual: No

3. After the incremental repository merge has been completed successfully, review the merge log files in `$SIEBEL_HOME\log`. See the following table for the Siebel Upgrade Wizard steps.

Step	Script or Input File	Description	Comments
1. Verify Siebel Version	verify_siebel_ver.sql	Verifies whether the correct Siebel CRM version (from which you are upgrading) was selected when the Siebel Database Configuration Wizard is started.	If this step stops because of an error, then it was because the wrong Siebel CRM version was selected during the launch of the Database Configuration Wizard. Select the Current Siebel Version as explained in the previous table.
2. Verify Repository Name	rename_existing_ repositories.sql	Renames the Siebel Repository to Prior Customer Repository.	None.
3. Drop Interface tables	dropif-db.sql	Removes all Siebel Enterprise Integration Manager tables.	None.
4. Drop triggers	trigdrop-db.sql	Removes all dynamically created triggers.	None.
5. Remove database-level functions and procedures	drop_db_func_ proc.sql	Removes the exchange rate function: exrate.	None.
6. Prepare for table creation	pret_irm.sql	Performs DDL operations, such as adding columns to tables. Performs DML operations, such as revising date formats.	None.
7. Create and update tables	ddlimp utility ddl.ctl as input	The ddl.ctl file specifies the structure of tables to be created or updated.	None.
8. Prepare for index creation	preschm_irm.sql	Performs DML operations. Moves data between tables. Changes the data in the existing fields, depending on the specified conditions.	None.
9. Create indexes	ddlimp utility ddl.ctl as input	The input file specifies the structure of indexes to be created.	None.

Step	Script or Input File	Description	Comments
10. Prepare prior customer repository	SWTClob.jar	This SWT to Object Definition utility converts Siebel Web Templates to object definitions in the Siebel Repository, eliminating the need to maintain and distribute external files to every server.	None.
11. Import seed data	dataimp utility seedupg0.inp as input seedupg1.inp as input seedupg_locale.inp as input	Before importing seed data, the dataimp utility deletes the existing seed data. The three seedupg files contain filters that the dataimp utility uses to prevent the deletion of seed data that you have modified or seed data that meets specified criteria. Unmodified seed data has a last update date (LAST_UPD) of 1980-01-01. The dataimp utility does not delete records in which the last update date (LAST_UPD) is later than this date.	None.
12. Upgrade data after seed data import	upg_data_afterseed.sql upg_data_afterseed_sia.sql	Updates existing seed data if any new columns have to be populated.	None.
13. Set system pref for Codepage DB	set_codepage.sql	Sets the database code page in the S_SYS_PREF table.	None.
14. Set system pref for unicode DB	set_unicode.sql	Sets the Unicode code page to UTF-8 in the S_SYS_PREF table.	None.
15. Update version component information	upd_upgcomp.sql	Updates the S_UPG_COMP table with the product release level. The S_UPG_COMP table stores version information for application executable programs.	None.
16. Run Oracle-specific DDL commands	ddlora.sql	Creates Oracle-specific DDL information, such as default storage parameters for docking objects, repository objects, and seed objects.	None.
17. Update Siebel database version	update_ver.sql seeduver.sql	The update_ver.sql script creates a temporary table, S_APP_VER_TEMP, which contains new version information for the database schema. The seeduver.sql script updates the S_APP_VER table with this information.	None.

Step	Script or Input File	Description	Comments
18. Encryption Upgrade	EncryptionUpgrade.j	Siebel Business Applications allow customers to encrypt sensitive information stored in the Siebel database, for example, credit card numbers, Social Security numbers, birth dates. This information cannot be viewed without access to Siebel Business Applications. Sensitive data can be encrypted by using AES (Advanced Encryption Standard). This utility identifies the RC2 encrypted columns and upgrades their data to the AES. Updates the logical layer of data for columns which are candidates for encryption.	None.
19. Export Prior Customer Repository	repimexp utility Export Prior Customer Repository	Exports the existing Prior Customer Repository to create New Customer Repository.	None.
20. Import Prior Siebel, New Siebel and New Customer repositories in parallel	repimexp utility Import the Prior Siebel Repository, New Siebel Repository, and New Customer Repository.	Imports the Prior Siebel Repository, New Siebel Repository, and New Customer Repository in parallel to the repository tables.	If this step produces an error while creating the New Customer Repository, check the log file in `SIEBEL_ROOT\siebsrvr\log\upgrade_path\output`. **For Oracle Database:** If you receive an error similar to the following, then you must restart the upgrade from the previous step *Export Prior Customer Repository* by changing the state.log of that step to Incomplete. `Error message: [DataDirect][ODBC Oracle driver][Oracle] ORA-12899: value too large for column "S_ACCELERATOR"."INA CTIVE_FLG" (actual: 4, maximum: 1)`
21. Install SQL packages	seeduver.sql	Verifies that correct versions are set in the S_APP_VER table.	None.
21. Install SQL packages (cont'd)	ifstrg.sql	Sets the storage parameters for Siebel Enterprise Integration Manager tables.	None.
21. Install SQL packages (cont'd)	ifindxstrg.sql	Sets the storage parameters for Siebel Enterprise Integration Manager table indexes.	None.
21. Install SQL packages (cont'd)	pkgseq.sql	Adds a suffix to row IDs in the S_SEQUENCE table. Ensures that row IDs are unique.	None.

Step	Script or Input File	Description	Comments
		The pkgldel.sql script defines s_ txn_log_del_proc. The procedure periodically deletes transactions from the S_DOCK_TXN_LOG file. It also deletes rows from the S_ DOCK_TXN SET table. This action prevents the need for a large rollback segment.	
21. Install SQL packages (cont'd)	trgreset.sql	Ensures that denormalized rows in the S_TERR table have the correct values.	None.
21. Install SQL packages (cont'd)	ddlseq.sql	Sets the sequence numbers for the specified tables.	None.
21. Install SQL packages (cont'd)	pkgvis.sql	Creates a function that modifies how Oracle Optimizer performs the visibility check.	None.
22. Create temporary indexes for merge performance	crt_temp_indexes_ merge.sql	Creates temporary indexes to improve the merge performance.	None.
23. Fix column alignment for custom objects	alignapplet.jar	Sets the standard for list column's header and data alignment based on type of fields they are mapped to across all the Siebel applications	None.
24. Take backup of the Siebel database and update database statistics	None	Pauses the Siebel Upgrade Wizard to enable a database backup and the execution of database statistics before the merge.	At this point in the upgrade, back up the database, and execute database statistics, then click Yes to continue the upgrade. To stop at this step and restart the wizard later, click No. To resume the upgrade, see *Preparing to Start the Siebel Upgrade Wizard*.
25. Execute Incremental Repository Merge	siebdev	Starts the incremental repository merge automatically for Windows. If you are using a Windows computer and Siebel Tools is installed for Siebel Enterprise servers in UNIX environment, then you must perform this step manually.	**For UNIX or Linux:** You must execute the incremental repository merge process manually. (On Microsoft Windows, incremental repository merge starts automatically.) See *Executing Incremental Repository Merge on UNIX*.
26. Generate Merge Report	MergeReport.jar	Generates the hierarchical merge report in HTML format. Also creates the `IRM_Merge*_ ERROR.txt` file in the `$SIEBEL_ HOME\log` folder, which contains	**For UNIX or Linux:** You must generate the hierarchical merge report manually, by running MergeReport.bat. Typically, you run this report after running incremental repository merge. (On Microsoft Windows, this report runs automatically.)

Step	Script or Input File	Description	Comments
		only the errors that are present in the **IRM_Merge*.txt** file. For more information about the hierarchical merge report, see *Reviewing the Hierarchical Merge Report*.	
27. Export Merge Data to CSV Script or Input File	ExportMrgCSV.jar	Exports repository-merge log data to a CSV file for business object, business component, applet, view, and Web page object definitions. This data can be compared with the Usage Pattern Tracking (UPT) CSV files to obtain the intersection data, so that you can focus testing on these objects only. The Siebel Upgrade Wizard exports the log data to CSV files in the following folder: **upgrep_log_directory/output/export_csv/csv** .	This step is executed only when you run IRM after migrating from a prior Siebel CRM release in which UPT was enabled (for example, version 15.5 or later, or version 16.0). For information about the specific releases (including applicable patchset releases) for which UPT can be enabled, see 2145521.1 (Article ID) on My Oracle Support. For more information about Usage Pattern Tracking, see *Siebel Applications Administration Guide* .
28. Record Upgrade History	store_history.sql	Stores the upgrade history in the S_INST_UPG_HIST table.	None.

Executing Incremental Repository Merge on UNIX

This task is a step in *Process of Upgrading Siebel Development Environment from Siebel CRM 8.1.1.x (SIA Repository)*. Use this task to execute incremental repository merge on UNIX.

Incremental repository merge is started automatically on Windows, but on UNIX you must initiate the process manually. You begin executing the incremental merge when the Upgrade Wizard pauses and notifies you to execute the manual merge.

To execute incremental repository merge on UNIX

1. Copy irm_sav_file.ssf from **$SIEBEL_ROOT\siebsrvr\bin** from the UNIX computer to **$SIEBEL_HOME\bin** on the Windows computer.
2. Execute the following command to start the incremental repository merge:

```
siebdev /u UserName /p Password /d "ToolsDataSource" /c "$SIEBEL_HOME\BIN\lang_code\tools.cfg"
/iPackmode /IRM UpgDeltaMerge
```

3. Click Yes to continue.

   To pause now and resume later, click No. To resume, see *Starting the Siebel Upgrade Wizard*.

   If the merge fails, see *Requirements for Restarting the Incremental Repository Merge*.

About Siebel Repository Merge Errors

This task is a step in *Process of Upgrading Siebel Development Environment from Siebel CRM 8.1.1.x (SIA Repository)*. Use this task to review Siebel Repository merge log files.

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

Acceptable and Unacceptable Errors

To determine whether the repository merge was successful, review the merge log files. The merge is successful if it completes without unacceptable errors:

- **Acceptable errors.** If an ancestor object is specified in an object definition, and if the ancestor object is not present in the New Siebel Repository, then a merge error occurs. This error is acceptable, and you can ignore it. The following is an example of an acceptable error in the merge log file, IRM_merge0.txt:

  ```
  !!ERROR::CANNOT upgrade objects which have Briefing Tracking Profile Applet - Product marked as 'Upgrade
   Anc'.
  ```

  If an object already exists in the New Customer Repository that has come from any Release Feature (RF) from Siebel CRM version 8.1.1.x or 8.2.2.x, then a merge error results stating that the object cannot be inserted because the same values already exist. This error is acceptable.

- **Unacceptable errors.** All other types of merge errors are unacceptable and indicate that the merge was not successful. For more information, see 477269.1 (Article ID) on My Oracle Support. This document was previously published as Siebel Troubleshooting Steps 19.
  Merge errors are displayed in the Upgrade Applications Objects List view in Siebel Tools. Additional details on merge errors are located in the incremental repository merge log file:

  ```
  $SIEBEL_HOME\log\IRM_merge0.txt
  ```

  Each time you run the merge process, the name of the IRM_merge0.txt file is incremented, for example, to IRM_merge1.txt.

Causes of an Incomplete Merge Process

If your repository merge process terminates and is flagged as Incomplete, then navigate to the Screens menu in Siebel Tools, and choose the Application Upgrader menu item. The most common reasons for its failure are as follows:

- The number of errors (!!ERROR) exceeds the number that was predefined in Siebel Tools when the merge was started.
- The merge process has been terminated because of a local issue on the Siebel Tools workstation, such as a scheduled reboot.
- RDBMS errors caused the process to stop.
- Memory allotment issues occurred on the workstation on which Siebel Tools is installed.
- A network failure occurred.

If the repository merge is terminated and flagged as Incomplete, then restart the incremental repository merge. For more information, see *Process of Meeting Requirements for an Incremental Repository Merge*.

Reviewing the Siebel Repository Merge Log Files

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

This task is a step in *Process of Upgrading Siebel Development Environment from Siebel CRM 8.1.1.x (SIA Repository)*. Use this task to review Siebel Repository merge log files.

To determine whether the repository merge was successful

1. From the Screens menu in Siebel Tools, choose Application Upgrader, and then Application Upgrade Object List.
2. In the Application Upgrades list, select the record of the merge.
3. Review the entry in the Status column:

    - **Completed.** Indicates the merge completed without errors.
    - **Completed with Errors.** Indicates the merge contains errors.

    If the Status column indicates Completed, then no further action is required. The merge was successful.

    If the Status column indicates Completed with Errors, then you must review the errors to determine whether the merge was successful. To review the errors, complete the remaining steps in this task.

    Note: These errors do not indicate an incomplete merge and rerunning the merge will not correct them.

4. In the Object Differences list, click Query.
5. In the Status field, enter the following: `!!ERROR::*`
6. Press Enter to run the query. A list of objects where the merge process encountered errors is returned.
7. Open the merge log file, IRM_merge0.txt, which is located in the following directory:

    ```
    $SIEBEL_HOME\log\IRM_merge0.txt
    ```

    If there are multiple files, then open the one with the highest number in the file name, for example, IRM_merge1.txt.
8. To locate merge errors in the file, search for the `!!ERROR` string. Informational messages are marked as `!!INFO`.
9. Use the objects displayed in the Object Differences list and the errors displayed in the log file to analyze the errors:

    - If all the errors are acceptable, then the merge is successful. It is advisable, however, to consider the number of acceptable errors when determining whether to rerun the merge operation.
    - If the log file contains unacceptable errors, then the merge has failed.

10. If the merge contains unacceptable errors, then see 477269.1 (Article ID) on My Oracle Support. This document was previously published as Siebel Troubleshooting Steps 19. This document explains many of the error messages that can appear in the log file. Use this document to correct the errors. If you cannot resolve all the errors, then contact Oracle Global Customer Support.
11. Open the workflow merge log file:

    ```
    $SIEBEL_HOME\log\IRM_merge0_ver.txt
    ```

    If there are multiple files, then open the file with the highest number in the file name, for example, IRM_merge1_ver.txt. This log file is created by the workflow premerge and postmerge steps.

12. Review the log file. If the file contains errors, then create a service request (SR) on My Oracle Support, or contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.

Reviewing the Hierarchical Merge Report

This task is a step in *Process of Upgrading Siebel Development Environment from Siebel CRM 8.1.1.x (SIA Repository)*. Use this task to review the hierarchical merge log files.

The hierarchical merge report is generated in HTML format for Application Hierarchy objects, which are Application, Screen, View, Business Component, Applet, Table, Business Service, Business Object and Workflow Process. This report allows you to review the tree of a particular object and see exactly what has been changed in the Application Hierarchy. If an object on which the currently selected object depends has changed, then you can drill down on that object and get a better idea of the testing effort involved.

The report is generated in the `SIEBSRVR_ROOT\log\upgrep_dev_*\output` directory and is named MergeReport.html.

When you run Incremental Repository Merge on Microsoft Windows, the HTML report is generated automatically once the incremental repository merge has completed. On UNIX, you must generate the hierarchical merge report manually, by running MergeReport.bat. Typically, you run this report after running incremental repository merge. For Upgrade, you also must generate the hierarchical merge report manually. MergeReport.bat is located in the `$SIEBEL_HOME\reppatch` directory.

You are prompted to specify values for the following parameters, which are required to generate the report. Where necessary, you must enclose parameter values in double-quotes.

- Tools Installation Folder
- ODBC Data Source Name
- Database Table Owner
- Database Table Owner Password
- Table Owner User Name
- Tablespace Name
- Index Space Name
- 16k Tablespace Name
- 32k Tablespace Name
- Is DB Unicode or Not
- Grantee Role (SSE_ROLE)
- Repository Name
- Full Path of Directory in Which to Generate HTML Report
- Full Path of Log File to Run Patch
- Database Platform (Oracle, MSSQL, DB2UDB, or DB2390)
- Merge ID for Report Generation. This value can be identified in Siebel Tools. Log in to Siebel Tools, navigate to Screens menu, then Application Upgrader, Application Upgrade Object List. Select a record and get the row ID from Help, About Record.
- Generate Report for Latest Merge Run. Enter Y to generate report. Enter N to display all available Merge Log ID.
- Retain Files for Debugging. Enter N to retain files for debugging. Otherwise, enter Y.

Usability

Different sections of the report present different types of information, as follows:

- The navigation section of the report, which appears in a side pane, lists the objects that are affected by the merge process. Select an object in this section to view information about the object.
- The upper or first section of the report lists the objects that are dependent on the object that is selected in the navigation section.
- The lower section of the report displays the attribute-level details of the object that is selected in the upper section.

Navigation Section

This report limits the objects that are part of the application hierarchy. Nine tabs are displayed for each object type. When a tab is selected, it expands and displays the list of objects that are affected by the merge process. Two options provide searching capability for any particular object in the report:

- Global Search searches any object in all objects of the application hierarchy.
- Local search limits the search functionality to (for example) a particular tab on which a search operation is triggered.

Dependent Object Section

The dependent object section, displays the dependent objects for the object that is selected in the navigation section. The selected object name is displayed before the dependent object section. You can further navigate to each tab to see the list of objects.

Tabs are displayed only for objects that are dependent on the selected object. For example, if Applet has been modified and no Business Service is dependent on that Applet, then the Business Service tab will not be displayed in this section. Under each tab, if an object name has no background color, it means that no attribute level information is changed as part of the merge for this object.

For example, for the View tab, responsibility information is displayed for the selected view. When the View tab is selected and a View object is selected, then the responsibility information for this view is also displayed.

The lower section of the report displays attribute-level information for the selected object. You can filter the resulting records based on Critical Conflicts, Non Critical Conflicts and All Changes.

Related Topic

Reviewing Siebel Repository Object Property Conflicts

Reviewing Siebel Repository Object Property Conflicts

This task is a step in *Process of Upgrading Siebel Development Environment from Siebel CRM 8.1.1.x (SIA Repository)*. Use this task to review Siebel Repository object property conflicts.

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

You can change how repository object property conflicts are resolved during the repository merge.

How Repository Object Property Conflicts Occur

Note: Use the Hierarchical Reports feature in Siebel Tools to learn what conflicts might have occurred, and to help identify areas to target for regression testing. For more information on this feature, see *Using Siebel Tools* .

You cannot resolve conflicts from the report, but you can use the report to identify where to target regression testing and what conflicts have occurred.

The repository merge compares repository object properties in the Prior Siebel Repository, Prior Customer Repository, and New Siebel Repository. When the value of an object property is different in all three repositories, an object property conflict exists. This conflict occurs when you have changed the value of an object property in the Prior Customer Repository, and the value of this property has also changed in the new release (New Siebel Repository).

An object property conflict does not occur if you changed the value of an object property in the Prior Customer Repository, and the object property value did not change in the new release. When a conflict does not happen, the merge process transfers the changed value to the New Customer Repository.

The merge process resolves object property conflicts by referring to the setting of the object's Standard Win property. For the majority of repository objects, the merge process resolves conflicts by using the object property value in the New Siebel Repository. Do not change the setting of the Standard Win property.

Reviewing Object Property Conflicts

You can review and change how object property conflicts were resolved using the Application Upgrade Attribute List view in Siebel Tools. The Attribute Differences List in the view includes the following columns:

- **Object Name.** The name of the object.
- **Attribute.** The object property name.
- **Conflict.** The merge process puts a check mark in this field if there was an object property conflict during the merge.
- **Resolution.** Displays which property value the merge process used to resolve the conflict:
  - **Standard Value.** The property value in the New Siebel Repository was used. This value is displayed in the New Standard column.
  - **Custom Value.** The property value in the Prior Customer Repository was used. This value is displayed in the Prior Customized column.
- **Override.** Puts a check mark in this column to change how the conflict is resolved. Overriding the resolution changes the property value in the merged repository. If the resolution was the Standard Value, then the displayed value is changed to the Custom Value and vice versa.

  Putting a check mark in the Override column does not change the value displayed in the Resolution column. A check mark indicates that the displayed value was manually overridden in the merged repository.
- **Prior Standard.** Displays the value of the object property in the Prior Siebel Repository.
- **Prior Customized.** Displays the value of the object property in the Prior Customer Repository. In the Resolution column, this value is called the Custom Value.
- **New Standard.** Displays the value of the object property in the New Siebel Repository. In the Resolution column, this value is called the Standard Value.
- **Conflict Resolution Type.** Displays whether the conflict is critical or non-critical. Only critical conflicts have to be resolved.

Related Topic

Reviewing the Hierarchical Merge Report

Reviewing Log Files for All Upgrades

This task is a step in *Process of Upgrading Siebel Development Environment from Siebel CRM 8.1.1.x (SIA Repository)*. The repository merge must have been successful. For information about reviewing repository merge log files, see *Reviewing the Siebel Repository Merge Log Files*. For information about reviewing log files, see *Reviewing the Siebel Upgrade Log Files*.

Reviewing Conflicts at the Attribute Level

This task is a step in *Process of Upgrading Siebel Development Environment from Siebel CRM 8.1.1.x (SIA Repository)*. Use this task to review conflicts at the attribute level.

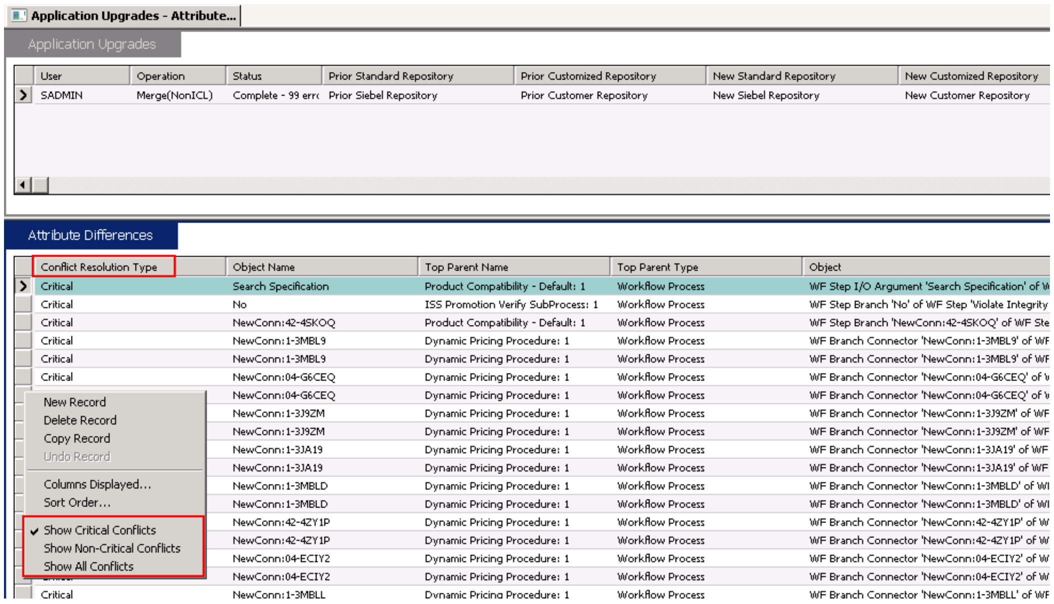
Platforms: Windows, UNIX, IBM z/OS.

For more information on review conflicts, see *Reviewing Siebel Repository Object Property Conflicts*.

To review conflicts at the attribute level

1. In Siebel Tools, from the Screens menu, choose Application Upgrader, and then Application Upgrade Attribute List.

   By default, you can see only the critical conflicts in the Application Upgrades Attribute differences view. You must review the critical conflicts. To view informational conflicts, right-click and select Show Non-Critical Conflicts. Noncritical conflicts include conflicts of attributes, such as Sequence number, Comments, Layout, and so on.



User	Operation	Status	Prior Standard Repository	Prior Customized Repository	New Standard Repository	New Customized Repository
SADMIN	Merge(NonICL)	Complete - 99 errc	Prior Siebel Repository	Prior Customer Repository	New Siebel Repository	New Customer Repository

Conflict Resolution Type	Object Name	Top Parent Name	Top Parent Type	Object
Critical	Search Specification	Product Compatibility - Default: 1	Workflow Process	WF Step I/O Argument 'Search Specification' of W
Critical	No	ISS Promotion Verify SubProcess: 1	Workflow Process	WF Step Branch 'No' of WF Step 'Violate Integrity
Critical	NewConn:42-4SKOQ	Product Compatibility - Default: 1	Workflow Process	WF Step Branch 'NewConn:42-4SKOQ' of WF Ste
Critical	NewConn:1-3MBL9	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:1-3MBL9' of WF
Critical	NewConn:1-3MBL9	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:1-3MBL9' of WF
Critical	NewConn:04-G6CEQ	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:04-G6CEQ' of W
	NewConn:04-G6CEQ	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:04-G6CEQ' of W
	NewConn:1-3J9ZM	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:1-3J9ZM' of WF
	NewConn:1-3J9ZM	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:1-3J9ZM' of WF
	NewConn:1-3JA19	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:1-3JA19' of WF
	NewConn:1-3JA19	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:1-3JA19' of WF
	NewConn:1-3MBLD	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:1-3MBLD' of WI
	NewConn:1-3MBLD	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:1-3MBLD' of WI
	NewConn:42-4ZY1P	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:42-4ZY1P' of W
	NewConn:42-4ZY1P	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:42-4ZY1P' of W
	NewConn:04-ECIY2	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:04-ECIY2' of W
	NewConn:04-ECIY2	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:04-ECIY2' of W
Critical	NewConn:1-3MBLL	Dynamic Pricing Procedure: 1	Workflow Process	WF Branch Connector 'NewConn:1-3MBLL' of WF

2. In the Application Upgrades list, select the record of the successful merge.
3. In the Attribute Differences list, click Query.
4. In the Attribute Differences list, click in the Conflict field so that a check mark appears.

5. Press Enter to run the query.

   The query displays a list of all object properties in which there is a conflict.
6. For each record, review the entry in the Resolution field.
7. To change the resolution, click in the Override field.

   A check mark appears that changes the value of the object property in the merged repository. Avoid overriding conflicts for the following object properties:
   - Left
   - Right
   - Top
   - Height
   - Width

   Review these properties in the upgraded application before changing them.

Marking Conflict Resolution as Complete Using Siebel Tools

This task is a step in *Process of Upgrading Siebel Development Environment from Siebel CRM 8.1.1.x (SIA Repository)*. Use this task to mark conflict resolution as Complete.

To mark conflict resolution as Complete using Siebel Tools

1. After you have resolved all conflicts, execute the following command from the `$SIEBEL_HOME\bin\siebdev.exe` directory on the command prompt:

```
$SIEBEL_HOME\bin\siebdev.exe /u SADMIN /p ******** /d <"ServerDataSrc">
/c "$SIEBEL_HOME\bin\enu\tools.cfg" /l enu /IRM UpgDeltaMerge /iPackmode
```

2. After Siebel Tools starts, select the Conflict Resolution Completed check box.
3. Click Finish.

   Clicking Finish creates an entry in the S_INST_UPG_HIST table, indicating that the merge and conflict resolution steps have been completed. If this step is not executed, then the next phase of upgrading the custom database schema (upgphys) produces an error in the first step.

Generating the Runtime Repository Data

For more information, see *Generating the Runtime Repository Data*.

Upgrading a Custom Database Schema

To upgrade a custom database schema, perform the following tasks:

- Execute the Siebel Database Configuration Wizard by specifying the following parameter values:

Parameter	Value
Environment Type	Development
Upgrade Options	Upgrade Custom Database Schema (upgphys)

  For more information about parameters, see *Preparing to Run the Siebel Database Configuration Wizard*.

  For more information about running the Database Configuration Wizard, see one of the following, depending on your platform:

  - *Running the Siebel Database Configuration Wizard on Windows*
  - *Running the Siebel Database Configuration Wizard on UNIX*

- Before running upgphys, Full Publish must be executed. For more information about Full Publish, see *Generating the Runtime Repository Data*.

Note: Before you upgrade a custom database schema using the upgphys process, it is strongly recommended to back up the entire database. If this upgrade operation fails, then you can restore the database to its preupgrade state, and rerun the upgrade operation.

The following table shows the different steps executed by the upgphys process. The Script or Input File column in the table lists the SQL file or input file that is executed in each step. The Description column provides a brief explanation of what the SQL file or input file does. The SQL files used for an upgrade and the contents of the SQL files vary depending on the upgrade mode and database.

Step	Script or Input File	Description	Solution
1. Check whether the merge and conflict resolution steps are complete	verify_irm.sql	Verifies whether the merge and conflict resolution steps are complete.	If this step stops because of an error, then the conflict resolution has not been completed. For information on resolving the conflict, see *Reviewing Siebel Repository Object Property Conflicts*. After you resolve the conflict, restart the Siebel Upgrade Wizard, and click No when asked if you want to continue. Then start the Siebel Upgrade Wizard again from the beginning. Oracle recommends verifying if the Full Publish logs were successful. For more information about Full Publish, see *Using Siebel Tools* .

Step	Script or Input File	Description	Solution
			CAUTION: Full Publish logs must be verified before running upgphys.
2. Drop temporary indexes created for merge performance	drop_temp_indexes_ merge.sql	Removes all the temporary indexes created to improve the merge performance.	None
3. Export schema definition	None	Exports the schema definition to the file `DBSRVR_ROOT/ platform/schema.ddl`. This file is used as an input to the production test and production environments.	None
4. Sync physical and logical tables	schema.ddl	Applies the schema.ddl and synchronizes the tables.	None
5. Sync physical and logical indexes	schema.ddl	Applies the schema.ddl and synchronizes the indexes.	None
6. Migrate query search controls of list applets into the Siebel database	SrchCntrlMigration.jar	This step migrates query search controls of list applets from files into the Siebel database. This step also exports the entire New Customer Repository to: `$DbsrvrRoot/ $DatabasePlatform/ custrep.dat`	None
7. Encryption Upgrade - Physical	EncryptionUpgrade.jar,	This step updates the physical layer data for columns which are encrypted with the RC2 algorithm. This updates the data to AES algorithm.	None.
8. Encryption Upgrade	EncryptionUpgrade.exe	Updates the physical layer data for columns which are encrypted with the RC2 algorithm. This updates the data to the AES algorithm.	None
9. Enable Workspace	EnableWorkspace.exe	Enables the Workspace on the New Customer Repository for repository content.	None
10. Export repository to a file	None	Exports the New Customer Repository to: `$DbsrvrRoot/ $DatabasePlatform/ custrep.dat`	None

Step	Script or Input File	Description	Solution
		This step exports the Design Time Repository (DR) to the custrep_dev.dat file and Runtime Repository (RR) to the custrep.dat file. The custrep.dat file contains only the required repository data and is used as an input to the production test and production environments.	
11. Import New Customer Repository as Siebel Repository	custrep_dev.dat	Truncates the repository tables and imports the previously exported New Customer Repository as Siebel Repository. This step imports the Design Time Repository (DR) from the custrep_dev.dat file.	None
12. Enable Workspace	EnableWorkspace.exe	Enables the Workspace on the New Customer Repository for Seed data (List of Values).	None
13. Migrate custom manifest data from XML files to the database **Note:** This step applies only to upgrades with custom Siebel Open UI manifest files for Siebel CRM version 8.1.1.9 or 8.1.1.10, and version 8.2.2.2 or 8.2.2.3.	• custom_manifest.xml • manifest_extensions.map	This step migrates Siebel Open UI custom manifest data from XML files to the database. For more information, see *About Migrating Siebel Open UI XML Manifest Data from Previous Release to New Manifest Data in the Database*.	None
14. Create list import package for Siebel Marketing	• mktg_listimp_pkg_spec.sql • mktg_listimp_pkg_body.sql	(Oracle Database only) This step creates a package to support the Siebel Marketing list import functionality for Oracle Database 12c. For more information, see Siebel Marketing Installation and Administration Guide.	None
15. Verify repository after upgrade	• dbchck • dictutl • mlovupgd • SeedConflict.jar	The repository is optionally verified after you complete the upgphys process. This step is executed if you selected the option Verify Repository After Upgrade in the Database Configuration Wizard. For more information, see *Verifying the Repository After Upgrade*.	None

Step	Script or Input File	Description	Solution
16. Upgrade history	store_history.sql	Stores the upgrade history in the S_INST_UPG_HIST table.	None

Verifying the Repository After Upgrade

As noted in Step 15 of the table in *Upgrading a Custom Database Schema*, the repository is optionally verified after you complete the upgphys process. This step is executed if you selected the option Verify Repository After Upgrade during database configuration. The following are the verification steps executed:

1. **dbchck.** Provides a report of the schema differences between the physical database and the repository definition. Unacceptable errors can occur if data types are mismatched. Acceptable errors occur if a schema object (such as a table or an index) is intentionally external to the repository.
2. **dictutl.** Verifies that all dock objects and rule definitions are correct.
3. **mlovupgd.** Validates the repository for data inconsistencies. Errors are logged into the log file and upgrade wizard continues with the next steps.
4. **SeedConflict.** Generates the seed conflict report in HTML format. The report contains details of all the records with unique constraint violations.

About Migrating Siebel Open UI XML Manifest Data from Previous Release to New Manifest Data in the Database

Platforms: Windows, UNIX, IBM z/OS.

In Siebel Open UI, before Siebel CRM 8.1.1.11 or 8.2.2.4, files to be downloaded for an applet were configured in one of the following ways:

- Through the Applet object properties Presentation_Model and Physical_Renderer in Siebel Tools.
- By adding entries under the Presentation Model and Physical Renderer sections in the manifest_extensions.map file in the `SIEBSRVR_ROOT\objects` folder on the Siebel Server.

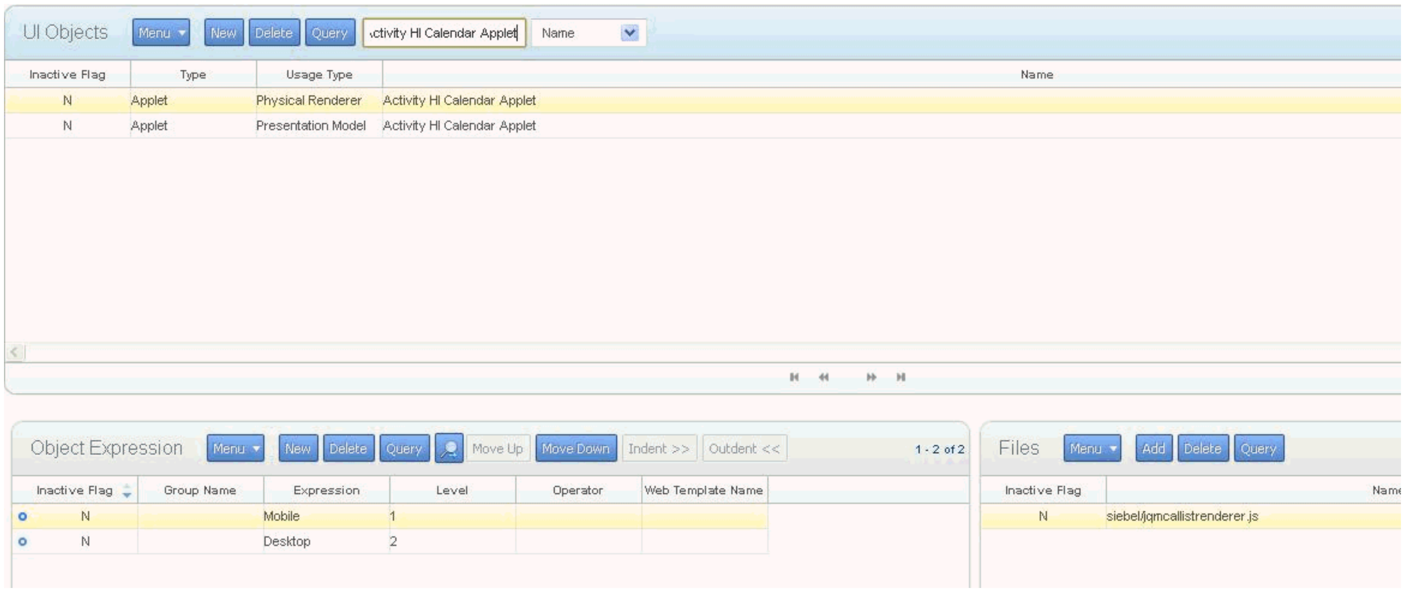
The files were then configured against the key in the custom_manifest.xml file or in an application-specific custom manifest file.

Starting with Siebel CRM 8.1.1.11 and 8.2.2.4, Siebel Open UI replaces the XML manifest file with manifest data that it stores in the Siebel database, as noted in *Upgrading a Custom Database Schema*. You cannot modify this predefined manifest data, but you can use the Manifest Administration screen in the client to configure the manifest data that your customizations require. Applets are mapped directly to files, and expressions are configured to determine under what conditions the applets use a given file.

The following example illustrates how data from the XML manifest file is migrated. The example uses the applet Activity HI Calendar Applet, which is used for calendar functionality in Siebel Open UI. It has separate renderings in mobile and desktop applications.

Note: The migration described in this topic applies only to upgrades with custom Siebel Open UI manifest files for Siebel CRM 8.1.1.9 or 8.1.1.10, and 8.2.2.2 or 8.2.2.3.

The following figure shows the UI Objects view with two Activity HI Calendar Applet UI object entries. One of these entries has a usage type of Renderer, while the other entry has a usage type of Presentation Mode. Two object expressions are shown in the Object Expression view: Mobile and Desktop, and in the Files view, the required JavaScript file is shown.



Inactive Flag	Type	Usage Type	Name
N	Applet	Physical Renderer	Activity HI Calendar Applet
N	Applet	Presentation Model	Activity HI Calendar Applet

Inactive Flag	Group Name	Expression	Level	Operator	Web Template Name
N		Mobile	1		
N		Desktop	2		

Inactive Flag	Name
N	siebel/jqmcallistrenderer.js

As the following example from an earlier version shows, the two separate renderings were implemented in the XML manifest file for the applet using the CalPR key:

```
<PLATFORM Name="IsMobileApplication()">
 <KEY Name=SignViewPM"
 <KEY Name="SignViewPR"
 <KEY Name="AppletPR"
 <KEY Name="CalPR">
 <FILE Name>siebel/jqmcallistrenderer.js</FILE_NAME>"
 </KEY>
 <KEY Name="GridPR"
 <KEY Name="RestOfUIPR"
 </PLATFORM>
 <PLATFORM Name="IsDesktopApplication()">
 <KEY Name="AppletPR">
 <KEY Name="ListPR">
 <KEY Name="GridPR">
 <KEY Name="TaskPanePR">
 <KEY Name="TreePR">
 <KEY Name="TileRenderer">
 <KEY Name="CalPR">
 <FILE Name>siebel/jqfullcalrenderer.js</FILE_NAME>"
 </KEY>
```

As shown in the example, separate files for mobile and desktop platforms are configured using the CalPR key. However, for the new manifest data in the Siebel database, there are two required configured entries: Mobile and Desktop. Both expressions correspond to a single user interface object, and they are evaluated sequentially by their level number. If the setting is True for either entry, then the respective file is downloaded, and further evaluation does not take place.

However, if a record is an Application type user interface object, with a usage type of Common, then all expressions set to True are considered, not just the first in Object Expression list. All True expressions are then aggregated and downloaded to the browser. For information on modifying manifest data, see Configuring Siebel Open UI.

Manifest File Migration Rules

The migration rules for this example are as follows. For each applet mapped to a manifest key in either of the following ways, an entry is made in the UI Objects applet.

- By using the Presentation_Model and Physical_Renderer user properties under the Applet object in Siebel Tools.
- By adding entries under the [Presentation Model] and [Physical Renderer] sections in the manifest_extensions.map file, in the *SIEBSRVR_ROOT* `\objects` folder on the Siebel Server.

Each time the manifest key is used in the legacy XML file, an entry is made in the new Manifest screen, and a record is also added to the Object Expression and Files views, using the rules shown in the following table.

Use of Keys in Legacy XML Files	Record Created in the Object Expression and Files Applets
Under a **`PLATFORM`** subsection within the **`PLATFORM_KEY_SPECIFIC`** section in the custom_manifest.xml	The record in the Object Expression applet with an expression value that evaluates to True only for the specified platform. The record also exists in the Files applet for any files associated with the record.
Under the **`KEY_COMMON`** section in the custom_manifest.xml	The record in the Object Expression applet with no entry in the Expression field. The record also exists the Files applet for any files associated with the record.
Under a **`PLATFORM`** section within the **`PLATFORM_KEY_SPECIFIC`** section in a custom Application Manifest XML file	The following records are created in the Object Expression applet: • A record for Group Expression • A record with an Expression value that evaluates to True only for the given platform • A record with an Expression value that evaluates to True only for the given application The records in the Files applet are associated with the Group Expression.
Under the **`KEY_COMMON`** section in a custom Application Manifest XML file	The record in the Object Expression applet with an Expression value that evaluates to True only for the given application and a record in the Files applet for the file associated with the record.

Default Configuration for Generic Applet Types and Configuration of Non-Applet User Interface Constituents

This topic applies to the following:

- **Applets that use the default configuration for generic applet types**. Such as List, Tree, and so on. In other words, applets that have not been mapped to any Manifest Key specific to an individual applet.
- **Other user interface elements.** Such as navigation user interface elements other than applets, whose keys are defined within the implementation itself, in other words, not through user properties or through the mapping file.

- **File downloads that occur during the application startup.**

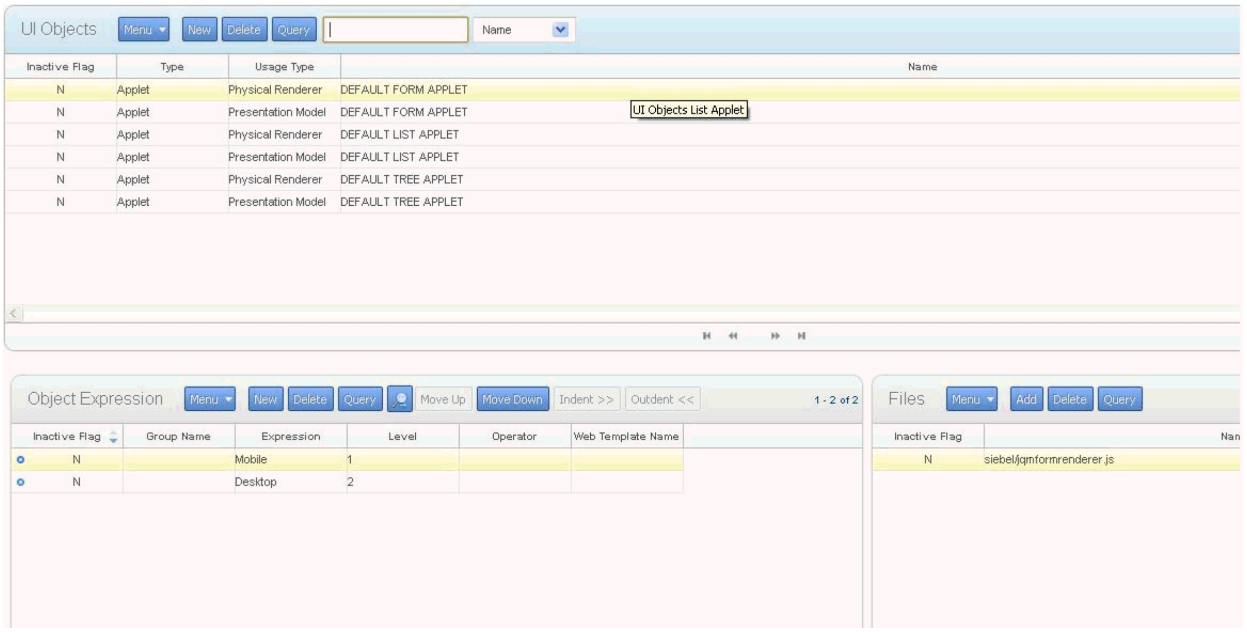
For generic applet types, the UI Object applet is populated with records with well known values for the Name field. The following table lists these values.

Key in the Legacy Manifest File	UI Object Type	UI Object Usage Type	Generic UI Object Name	Description
AppletPM	Applet	Presentation Model	DEFAULT FORM APPLET	Key from the default Form Applet Presentation Model.
AppletPR	Applet	Physical Renderer	DEFAULT FORM APPLET	Key for the default Form Applet Physical Renderer.
ListPM	Applet	Presentation Model	DEFAULT LIST APPLET	Key for the default List Applet Presentation Model.
ListPR	Applet	Physical Renderer	DEFAULT LIST APPLET	Key for the default List Applet Physical Renderer.
TreePM	Applet	Presentation Model	DEFAULT TREE APPLET	Key for the default Tree Applet Presentation Model.
TreePR	Applet	Physical Renderer	DEFAULT TREE APPLET	Key for the default Tree Applet Physical Renderer.
ScreenPM	Navigation	Presentation Model	NAVIGATION CONTROL	Key for the default Navigation Control Presentation Model.
RestOfUIPM	Application	Common	PLATFORM INDEPENDENT	Key for Presentation Model implementation of parts of the user interface other than those with specific keys for their default implementation, such as Menu, Toolbar, and so on.
RestOfUIPR	Application	Common	PLATFORM DEPENDENT	Key for Physical Renderer implementation of parts of user interface other than those with specific keys for their default implementation, such as Menu, Toolbar, and so on.
NAVIGATION_TREE	Navigation	Physical Renderer	NAVIGATION TREE	Key for the default tree Navigation Control Physical Renderer.
NAVIGATION_TAB	Navigation	Physical Renderer	NAVIGATION TAB	Key for the default tab Navigation Control Physical Renderer.

Generic Applet Types from the XML Manifest File

Platforms: Windows, UNIX, IBM z/OS.

The example of a generic Mobile applet shown in the following image uses the AppletPR key in the XML manifest file. This key has two entries in the Object Expression view: Mobile and Desktop. The Files view contains the JavaScript file (siebel/jqmformrenderer.js in this example) that the manifest file maps to display the applet.

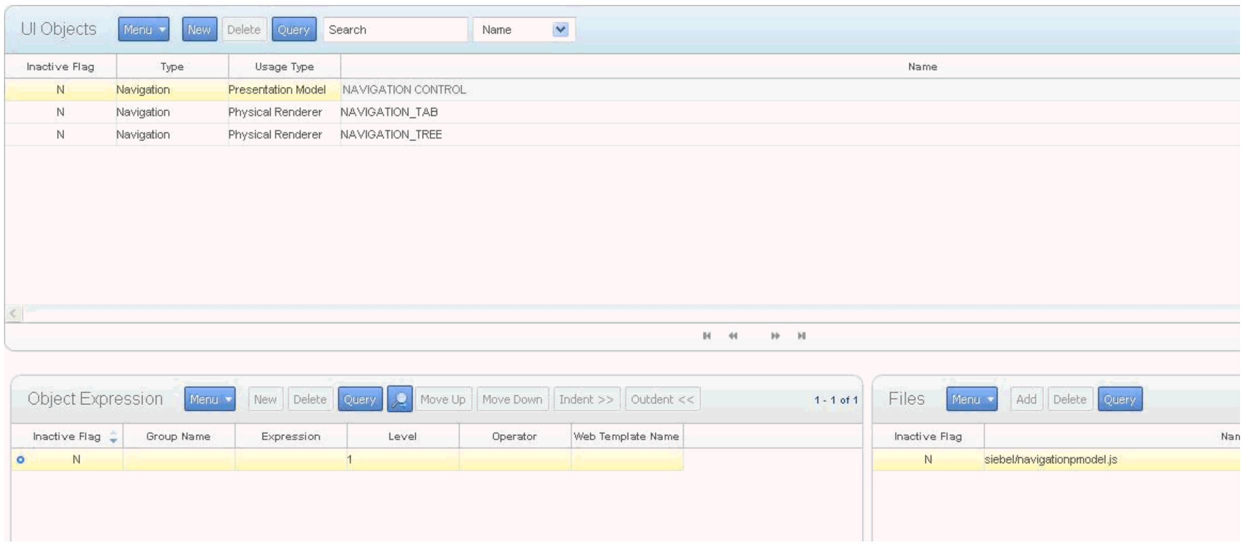


Inactive Flag	Type	Usage Type	Name
N	Applet	Physical Renderer	DEFAULT FORM APPLET
N	Applet	Presentation Model	DEFAULT FORM APPLET
N	Applet	Physical Renderer	DEFAULT LIST APPLET
N	Applet	Presentation Model	DEFAULT LIST APPLET
N	Applet	Physical Renderer	DEFAULT TREE APPLET
N	Applet	Presentation Model	DEFAULT TREE APPLET

Inactive Flag	Group Name	Expression	Level	Operator	Web Template Name
N		Mobile	1		
N		Desktop	2		

Inactive Flag	Nan
N	siebel/jqmformrenderer.js

The following image shows the entries for the navigation controls, such as ScreenPM, in the UI Objects screen. The Files view contains the following file: `siebel/navigationmodel.js`.

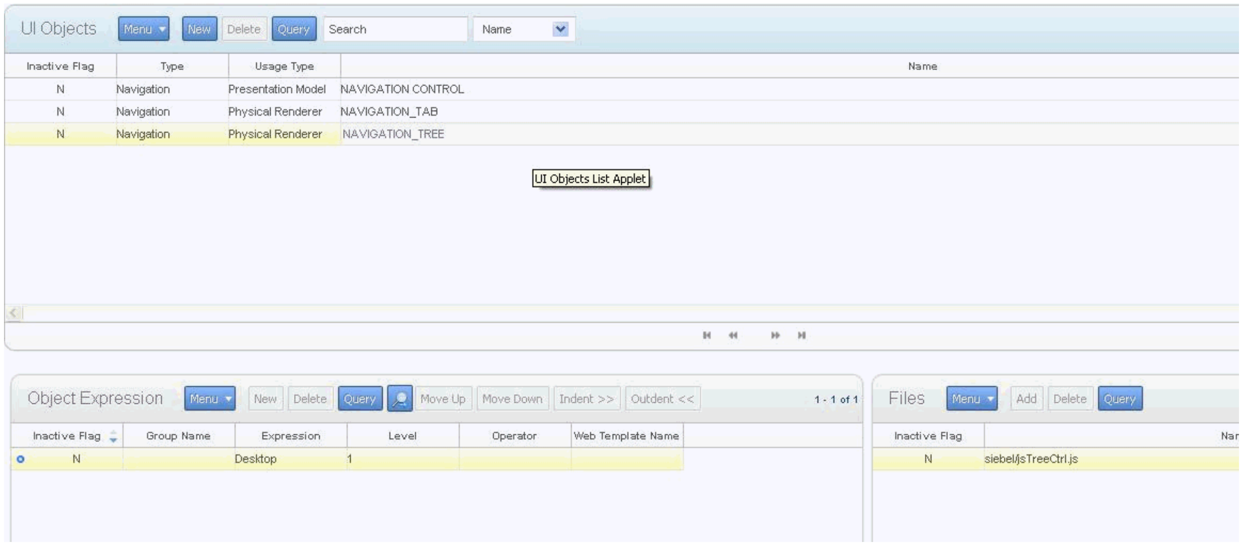


Inactive Flag	Type	Usage Type	Name
N	Navigation	Presentation Model	NAVIGATION CONTROL
N	Navigation	Physical Renderer	NAVIGATION_TAB
N	Navigation	Physical Renderer	NAVIGATION_TREE

Inactive Flag	Group Name	Expression	Level	Operator	Web Template Name
N			1		

Inactive Flag	Nan
N	siebel/navigationpmodel.js

The following image shows the entries for NAVIGATION_TREE. There is one entry in the Object Expression view: Desktop. The Files view contains the following file: `siebel/jsTreeCtrl.js`.

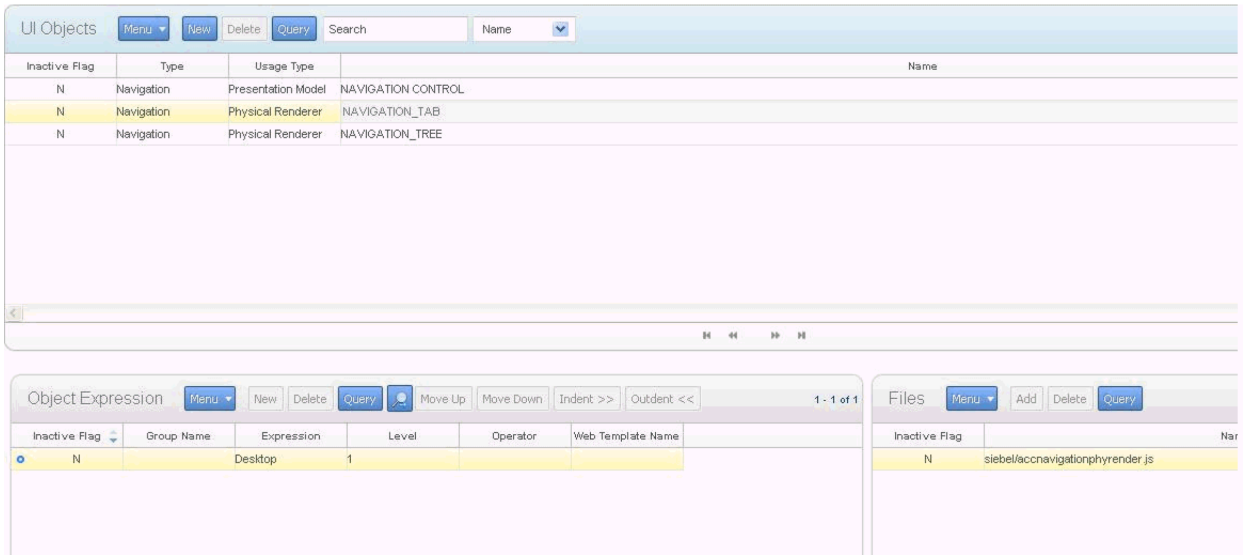


Inactive Flag	Type	Usage Type	Name
N	Navigation	Presentation Model	NAVIGATION CONTROL
N	Navigation	Physical Renderer	NAVIGATION_TAB
N	Navigation	Physical Renderer	NAVIGATION_TREE

Inactive Flag	Group Name	Expression	Level	Operator	Web Template Name
N		Desktop	1		

Inactive Flag	Nar
N	siebel/jsTreeCtrl.js

The following image shows the entries for NAVIGATION_TAB. There is one entry in the Object Expression view: Desktop. The Files view contains the following file: **`siebel/accnavigationphyrender.js`**.

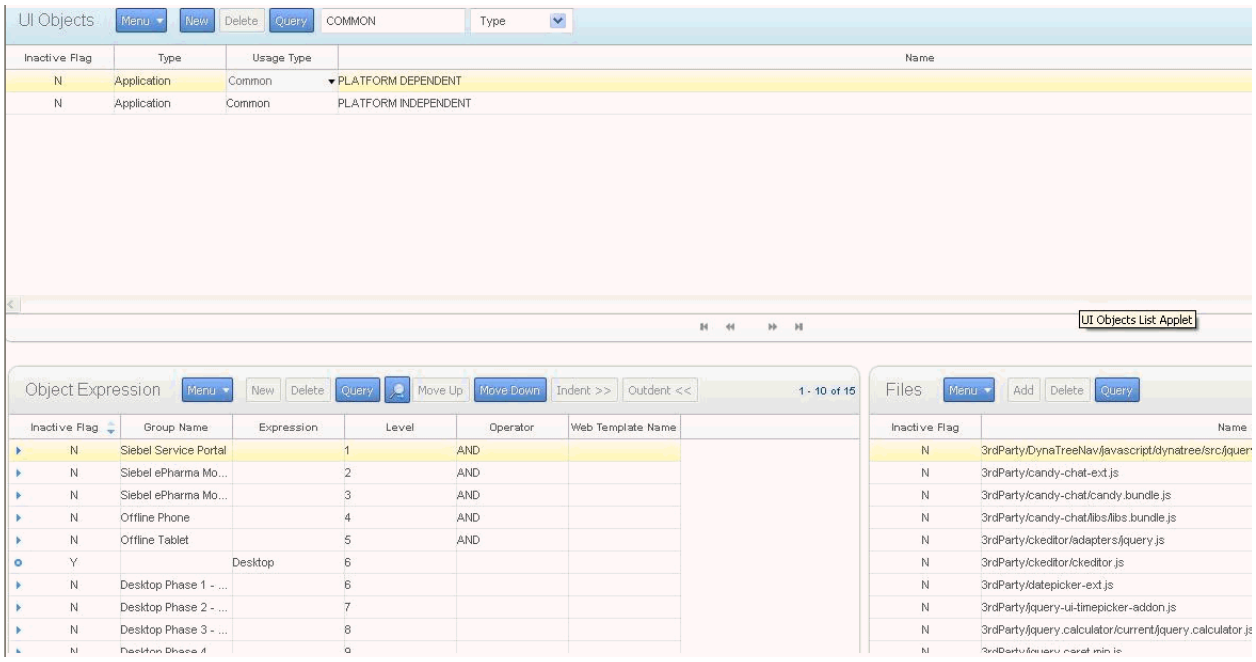


Inactive Flag	Type	Usage Type	Name
N	Navigation	Presentation Model	NAVIGATION CONTROL
N	Navigation	Physical Renderer	NAVIGATION_TAB
N	Navigation	Physical Renderer	NAVIGATION_TREE

Inactive Flag	Group Name	Expression	Level	Operator	Web Template Name
N		Desktop	1		

Inactive Flag	Nar
N	siebel/accnavigationphyrender.js

The files specified in the legacy manifest keys, RestOfUIPM and RestOfUIPR, are now located within the entries for UI Objects named PLATFORM_INDEPENDENT and PLATFORM_DEPENDENT. The following image shows the migrated legacy manifest keys.



Inactive Flag	Type	Usage Type	Name
N	Application	Common	PLATFORM DEPENDENT
N	Application	Common	PLATFORM INDEPENDENT

Inactive Flag	Group Name	Expression	Level	Operator	Web Template Name
N	Siebel Service Portal		1	AND	
N	Siebel ePharma Mo...		2	AND	
N	Siebel ePharma Mo...		3	AND	
N	Offline Phone		4	AND	
N	Offline Tablet		5	AND	
Y		Desktop	6		
N	Desktop Phase 1 - ...		6		
N	Desktop Phase 2 - ...		7		
N	Desktop Phase 3 - ...		8		
N	Desktop Phase 4		9		

Inactive Flag	Name
N	3rdParty/DynaTreeNav/javascript/dynatree/src/jquer
N	3rdParty/candy-chat-ext.js
N	3rdParty/candy-chat/candy.bundle.js
N	3rdParty/candy-chat/libs/libs.bundle.js
N	3rdParty/ckeditor/adapters/jquery.js
N	3rdParty/ckeditor/ckeditor.js
N	3rdParty/datepicker-ext.js
N	3rdParty/jquery-ui-timepicker-addon.js
N	3rdParty/jquery.calculator/current/jquery.calculator.js
N	3rdParty/jquery.caret.min.js

Files configured against a number of sections in the legacy manifest XML file are now part of these two UI Objects:

- PLATFORM DEPENDENT
- PLATFORM INDEPENDENT

The following table shows the mapping of the legacy XML file.

Legacy XML Section	Legacy Manifest Specific Subsection	Legacy Manifest Key	UI Object Name	UI Object Expression
COMMON	Not applicable	Not applicable	PLATFORM INDEPENDENT	Not applicable
PLATFORM_ COMMON	PLATFORM Name="IsDesktopApplication()"	Not applicable	PLATFORM DEPENDENT	Desktop
PLATFORM_KEY_ SPECIFIC	PLATFORM Name="IsDesktopApplication()"	Not applicable	PLATFORM DEPENDENT	Desktop
KEY_COMMON	Not applicable	RestOfUIPM	PLATFORM INDEPENDENT	Not applicable
PLATFORM_KEY_ SPECIFIC	PLATFORM Name="IsDesktopApplication()"	RestOfUIPR	PLATFORM DEPENDENT	Desktop

Note: When processing Object Expressions Records to choose files for a given UI Object, the first Expression that evaluates to True is selected, and the file associated with it is downloaded to the browser. However, there is one exception. With the UI Object records of the Application type and Common usage type, all expressions in the Object Expression list are considered, and then they are aggregated and downloaded to the browser.

Differences Between Customer Manifest Data and Oracle Manifest Data

All manifest data that replaces the previous configuration in the core_manifest.xml file is seeded in the database that is packaged with the release of Siebel CRM version. You cannot change the Oracle manifest seed data. For any UI Objects record that has been seeded in the database by Oracle, the record itself and all child Object Expression records are read-only.

If you want to customize a given UI Object, then a separate high-level UI Objects record must be created, and the required Object Expression records must be created under it. For example, if a UI Object record with a Name value of Contact List Applet is seeded in the manifest database for the Usage Type Physical Renderer, and if you want to override the configuration with a different implementation of the Physical Renderer, then you must create a new UI Objects record with a Name value of Contact List Applet and a Usage Type value of Physical Renderer, and the Object Expression record you want to create and corresponding Files record must be entered.

Note: If you add a UI Objects record with the same Name, Type, and Usage Type values to a UI object that is already seeded in the Manifest database, then your newly created record is given precedence by the Manifest Runtime engine, and is used when evaluating true Object Expression values when choosing a File record to download. There is one exception, however. For UI Objects records with a Type value of Application and a Usage Type value of Common, both the Oracle seeded record and your new record are considered. The sum total of all the chosen Files for both the seeded record and your new record is aggregated and downloaded to the browser.

File Selection Rules

The following table shows file selection rules.

UI Object Record Values	Oracle (Siebel?) Record Present	User Created Record	File Selection Rule
Type = 'Application' and Usage Type = 'Common'	Yes	No	All files against all winning expressions under the Oracle supplied UI Objects record are aggregated and downloaded to the browser.
Type = 'Application' and Usage Type = 'Common'	Yes	Yes	All files against all winning expressions under both the Oracle supplied and user entered UI Objects record are aggregated and downloaded to the browser.

UI Object Record Values	Oracle (Siebel?) Record Present	User Created Record	File Selection Rule
Usage Type != 'Common'	Yes	No	The file against the first winning expressions under the Oracle supplied UI Objects record is downloaded to the browser.
Usage Type != 'Common'	Yes	Yes	The file against the first winning expressions under the user entered UI Objects record is downloaded to the browser.

Checking the Post-upgphys Verification Process

After the upgphys upgrade is completed, check the oui_manifest.log file located in `SIEBEL_ROOT\siebsrvr\upgphys_dev_versionnumber\output` to verify whether the upgrade was successful.

Checking for Database Errors

Use the following method to check for database errors.

To check for database errors

- If you encounter database errors during the running of the upgphys upgrade, check the manifest_sqllog.txt file.

Checking for Application Errors

Use the following method to check for application errors.

To check for application errors

- If you encounter application errors during the running of the upgphys upgrade, check the oui_manifest.txt file.

Process of Regenerating the Siebel Repository Definition Files

The tasks in this process enable you to regenerate the Siebel Repository definition files.

Environments: Development environment only.

If you have modified repository objects after the development environment upgrade (upgphys) and before upgrading the production test environment, then you must regenerate the schema.ddl and custrep.dat files. These files were created during the development environment upgrade (upgphys):

- **Schema.ddl.** This file contains the logical definition of the Siebel database.
- **Custrep.dat.** This file contains the definition of the repository objects.

These files are used as input to the production test and production environment upgrades. These files contain only the Runtime Repository and required Design Time Repository which is required for production environments. There is an additional custrep_dev.dat that contains the entire development Repository. This can be created for backup. If you modify the object definitions or the schema definitions in the repository after these files have been created, you must regenerate the files. Perform the following tasks to regenerate the Siebel Repository definition files:

1. *Regenerating the schema.ddl File*
2. *Regenerating the custrep.dat File*
3. *Editing the Siebel Tools Configuration File After the Development Environment Merge*

Regenerating the schema.ddl File

This task is a step in *Process of Regenerating the Siebel Repository Definition Files*. Use this procedure to regenerate the schema.ddl file.

To regenerate the schema.ddl file

1. On the Siebel Server where the Siebel Database Server files are installed, navigate to the following location:

   - **Windows.** `SIEBEL_ROOT\bin`
   - **UNIX.** `$SIEBEL_ROOT/bin`

2. Run the following command:

   ```
   ddldict /u DatabaseOwner /p Password /c "ODBCDataSource" /d TableOwner
   /f DBSRVR_ROOT\DatabasePlatform\schema.ddl /e y /a y /l SiebelLogDir\sch_dict.log
   /n "Siebel Repository" /t dcir
   ```

   where:

   - `DatabaseOwner` is the Siebel database administrator account name.
   - `Password` is the Siebel database administrator account password.
   - `ODBCDataSource` is the ODBC name for connecting to the database. Enclose the name in quotation marks.
   - `TableOwner` is the Siebel table owner name.
   - `DBSRVR_ROOT` is the absolute path to the Siebel Database Server installation directory.
   - `DatabasePlatform` is the Siebel Database Server directory name for the database, for example, Oracle. The example shows the Windows path syntax. On UNIX systems, use the UNIX path syntax.
   - `SiebelLogdir` is the path to the directory where you want the output log placed; that is, the log output directory. The example shows the Windows path syntax. On UNIX systems, use the UNIX path syntax.

3. After the command completes, review the output log files for errors. If the log indicates there are errors, then create a service request (SR) on My Oracle Support, or contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.

Regenerating the custrep.dat File

This task is a step in *Process of Regenerating the Siebel Repository Definition Files*. Use this procedure to regenerate the custrep.dat file.

To regenerate the custrep.dat file

1. On the Siebel Server where the Siebel Database Server files are installed, navigate to the following location:

   Windows: `SIEBEL_ROOT\bin`

   UNIX: `$SIEBEL_ROOT/bin`
2. Run the following command:

   ```
   repimexp /a O /u DatabaseOwner /p Password /c "$ODBCDataSource" /d TableOwner
   /r "Siebel Repository" /f $DbsrvrRoot/$DatabasePlatform/custrep.dat /l $SiebelLogDir/exprep.log
   /v Y /Y "SrcWorkspaceName=MAIN;SrcEndVer=;Lang=ALL"
   ```

   where:

   - `DatabaseOwner` is the Siebel database administrator account name.
   - `Password` is the Siebel database administrator account password.
   - `ODBCDataSource` is the ODBC name for connecting to the database. Enclose the name in quotation marks.
   - `TableOwner` is the Siebel table owner name.
   - `DBSRVR_ROOT` is the absolute path to the Siebel Database Server installation directory. The example shows Windows path syntax. On UNIX systems, use UNIX path syntax.
   - `DatabasePlatform` is the Siebel Database Server directory name for the database, for example, Oracle.
   - `SiebelLogdir` is the path to the directory where you want the output log placed (log output directory). The example shows Windows path syntax. On UNIX systems, use UNIX path syntax.

     This command exports only the repository content that is required on the production environment. The entire repository is not needed on the production environment for Siebel CRM 17.0 and later releases.
3. After the command has finished, review the output log files for errors. If the log indicates there are errors, then create a service request (SR) on My Oracle Support, or contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.

Editing the Siebel Tools Configuration File After the Development Environment Merge

This task is a step in *Process of Regenerating the Siebel Repository Definition Files*. Use this task to edit the Siebel Tools configuration file after successfully performing your Development environment repository merge.

CAUTION: You must set the prefix to back to X_ from changed value of SBL_ to enable you, after the merge, to create your own custom symbolic strings.

To change the Sym Str Prefix value in tools.cfg file

1. Navigate to the `$SIEBEL_HOME\bin\<lang_code>` directory, and open the tools.cfg file.

2. Make the following changes in the tools.cfg file:

   a. Change the SymStrPrefix value from SBL_ back to the default value of X_.
   b. Make sure the Set EnableToolsConstrain value is set to the default value of False. If it is not, then set the value to False.
   c. Change the DockRepositoryName value from New Customer Repository to Siebel Repository.
3. Save the tools.cfg file.

Performing a Production Test or Production Environment Migration from Siebel CRM 8.1.1.x (SIA Repository)

Environments: Production test, production.

To upgrade your production environment use the following task. For information on Siebel Database Configuration Wizard parameters, see *Preparing to Run the Siebel Database Configuration Wizard*. For information on running the Database Configuration Wizard, see *Running the Siebel Database Configuration Wizard on Windows* or *Running the Siebel Database Configuration Wizard on UNIX*, depending on your platform.

Requirements: You must copy the schema.ddl and the custrep.dat files to your production test environment and production environment to conduct upgrades in each of these environments. For more information, see *Process of Regenerating the Siebel Repository Definition Files*.

To perform a production upgrade

1. In the Siebel Database Configuration Wizard, select the following options:

   - Environment Type: Production
   - Upgrade Options: Upgrade Custom Database Schema (upgrep and upgphys)
2. Start the Siebel Upgrade Wizard.

   For information on running the Siebel Upgrade Wizard, see *Preparing to Start the Siebel Upgrade Wizard*.

   The Siebel Upgrade Wizard reads the steps from the driver file that relate to your Siebel CRM version. The Siebel Upgrade Wizard performs the commands listed in those steps.

Note: When run in production mode, the Siebel Upgrade Wizard does not ask for the location of web templates and JS scripts like it does when run in development mode. This is because custom web templates are stored directly in the database for production/test environments. Also, any changes are made in the development environment first and then propagated to production/test environments. As a result, custom scripts references are present in the manifest tables.

16 Siebel Postmerge Development Tasks

Siebel Postmerge Development Tasks

This chapter provides guidelines for developers following a successful Siebel Repository merge. This chapter includes the following topics:

- *Reviewing Objects Deleted from the Siebel Repository*
- *Reviewing Obsolete Objects in the Siebel Repository*
- *Upgrading to the Siebel Symbolic String Model*
- *Dropping IBM DB2 8-KB Tablespaces and Buffers After a Siebel Merge*
- *Updating Siebel Enterprise Application Integration (EAI)*

Reviewing Objects Deleted from the Siebel Repository

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

If an object you have deleted from the Prior Customer Repository exists in the New Customer Repository, then the repository merge does not delete the object from the New Customer Repository. After the merge, you must review these objects and verify that they do not adversely affect the operation of the application.

Requirements: The repository merge must have been successful. See *Reviewing the Siebel Repository Merge Log Files*.

To generate a list of deleted objects

1. In Siebel Tools, from the Screens menu, choose Application Upgrader, and then Application Upgrade Object List.
2. In the Application Upgrades list, select the record of the successful merge.
3. Click Query.
4. Enter your query criteria in the Object Differences list:
   - Click in the In Prior Standard field so that a check mark appears.
   - Click in the Added to New Customized field so that a check mark appears.
   - Click in the In Prior Customized field so that a check mark appears. Then click in it again so that no check mark appears.
5. Press Enter to run the query.

   Deleted objects appear in the Object Differences list. You can filter the objects displayed by using the Top Parent Type and Object Type fields.
6. Review the list carefully to determine that deleted objects that have been restored to the merged repository will not have an adverse effect on upgraded applications.

Related Topic

About the Siebel Repository Merge

Reviewing Obsolete Objects in the Siebel Repository

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

Objects that were available in the Prior Siebel Repository but are not available in the New Siebel Repository are obsolete. After performing the repository merge, you can generate a list of obsolete objects using Siebel Tools. Objects that were available in the Prior Siebel Repository are compared with the objects that are available in the New Siebel Repository.

Requirement: The repository merge must have been successful. See *Reviewing the Siebel Repository Merge Log Files*.

To generate a list of obsolete objects

1. In Siebel Tools, from the Screens menu, choose Application Upgrader, and then Application Upgrade Object List.
2. In the Application Upgrades list, select the record of the successful merge.
3. Click Query.
4. Enter your query criteria in the Object Differences list:
    a. Click in the In Prior Standard field so that a check mark appears.
    b. Click in the Added or New Standard field so that a check mark appears.
    c. Click in the In Prior Customized field so that a check mark appears. Then click in the field again so that no check mark appears.
    d. Click in the Attribute field so that a check mark appears. Then click in the field again so that no check mark appears.
5. Press Enter to run the query.

    Deleted objects appear in the Object Differences list. You can filter the objects displayed by using the Top Parent Type and Object Type fields.
6. Review the list carefully to determine that obsolete objects that have been deleted will not have an adverse effect on upgraded applications.

Related Topic

About the Siebel Repository Merge

Upgrading to the Siebel Symbolic String Model

Environments: Development environment only.

The symbolic string model is object-oriented. A single symbolic string replaces the text string translations. For each language, a text string is defined and assigned to the symbolic string as an attribute. This simplifies multilingual management of text strings throughout the user interface.

Some strings will not be converted to the symbolic string model during upgrade. Seed data, error messages, lists of values (LOVs), and non-translatable attributes (such as the text alignment property on a control) will continue to use locale-based strings.

You must execute a conversion utility (consoleapp) to convert and consolidate your custom locale-based strings to the new model. If you plan to install a language pack, then it is recommended that you do so before you run the string conversion or consolidation process.

Instructions for converting or consolidating to the symbolic strings model are found in *Using Siebel Tools* .

Related Topics

About the Siebel Repository Merge

Upgrade Planning for Siebel String Translation

Dropping IBM DB2 8-KB Tablespaces and Buffers After a Siebel Merge

Environments: Development environment only.

Databases: IBM DB2 UDB only.

Drop the 8-KB tablespace, 8-KB temporary tablespace, and 8-KB bufferpool. Before dropping your 8-KB tablespace, check for the existence of any tables in it by running the following SQL statement:

```
select name from sysibm.systables where TBSPACE='TBS_8K'
```

Updating Siebel Enterprise Application Integration (EAI)

Environments: Development environment only.

Platforms: Windows and UNIX only.

If you use Enterprise Application Integration (EAI), then perform the following procedure to update the definitions of the Business Objects to account for changes in data type, length, edit format or other properties.

To upgrade integration objects

1. Determine whether you need to synchronize the integration objects, and synchronize if necessary.

   To determine whether you need to synchronize integration objects, review the synchronization considerations in Integration Platform Technologies: Siebel Enterprise Application Integration.
2. Validate the integration objects.

3. If you receive validation errors, then deactivate the user keys or fields that cause the error.
4. If you receive the error *List Of* in the XML Parent Element, then manually remove the value `List of` from the XML Parent Element.

17 Postupgrade Tasks for the Siebel Database

Postupgrade Tasks for the Siebel Database

This chapter provides tasks to be completed within the Siebel database, and the Siebel File System following a successful upgrade and repository merge. This chapter includes the following topics:

Reapplying Schema Customizations in the Siebel Database

Environments: Development environment only.

In the current release, tables are obsolete or have been replaced by new tables. If you added extension columns or foreign key (FK) columns to tables that are obsolete in the current release, then you must reapply these changes to the new tables.

Reviewing Obsolete Tables

The upgrade process generates a report that you can review for information about tables that are either obsolete. This report, `xtndobstbl.txt`, lists the following:

- Custom columns in obsolete tables
- Custom foreign key columns pointing to obsolete tables
- Siebel Enterprise Integration Manager mappings for custom foreign key columns to access-control related obsolete tables
- Workflow columns by custom foreign key to obsolete tables
- Customer denormalized columns on obsolete tables
- Obsolete tables in the current release

Each obsolete table is listed with one of three codes:

- **Not Used.** These tables are not used in the current release, but you can continue to use them. These tables are supported as is (for instance, with docking or Siebel Enterprise Integration Manager).
- **EOL (end of life).** These tables are not used in the current release, and they are not supported in future releases.

- **Inactive.** These tables have been discontinued, and are not supported in the current release. You must move extension columns and foreign key columns that reside on inactive tables to alternate tables.

If no tables are listed in `xtndobstbl.txt`, then no action is required.

If this file lists any tables, then reapply the custom extensions and foreign key columns to those tables in the current release using Siebel Tools. See *Configuring Siebel Business Applications* .

Migrating Address Data After a Direct SEA to SIA Upgrade

Upgrades from: SEA repository to SIA repository.

Platforms: Windows and UNIX only.

See the following table for a list of schema differences between the SEA and SIA repository along with recommended data migration approaches.

Schema Differences Between SEA and SIA Repositories

The following table displays schema differences between the SEA repository and the SIA repository.

SEA Repository	SIA Repository	SEA to SIA Migration Approach
Account Addresses are stored in the S_ ADDR_ORG table. Other addresses are stored in the S_ADDR_PER table.	The S_ADDR_ORG table is obsolete. Address data is stored in the S_ADDR_PER table.	Data is converted automatically through upgrade scripts.
Organization Relationships are stored in the S_ORG_REL table. Other party relationships are stored in the S_PARTY_REL table.	All party relationships, including organizations, are stored in S_PARTY_REL. The S_ORG_REL table is obsolete.	Manual data migration required. For instructions see 476479.1 (Article ID) on My Oracle Support. This document was previously published as Siebel Technical Note 312.
Contact relationships are stored in the S_CONTACT_REL table. Other party relationships are stored in the S_PARTY_REL table.	All party relationships, including organizations, are stored in the S_PARTY_ REL table. The S_CONTACT_REL table is obsolete.	Manual data migration required. For instructions see 476479.1 (Article ID) on My Oracle Support. This document was previously published as Siebel Technical Note 312.
Household relationships are stored in the S_ PER_ORG_UNIT table.	Household relationships are stored in the S_ PARTY_PER table.	Manual data migration required. For instructions see 476479.1 (Article ID) on My Oracle Support. This document was previously published as Siebel Technical Note 312.

Validating Dock Objects and Rule Definitions in the Siebel Database

Environments: Development, production test, production.

Platforms: Windows, UNIX, IBM z/OS.

Changes to visibility rules and dock objects require the assistance of Oracle Advanced Customer Services.

If you deploy Siebel Business Applications to mobile users with local databases, then you can run the `DICTUTL` utility to verify that all dock objects and rule definitions are correct. Dock objects allow mobile users to synchronize their local databases with the Siebel Server. Rules determine which data users synchronize. For more information about dock objects and rules, see Siebel Tools Online Help and *Siebel Remote and Replication Manager Administration Guide* .

To verify that all dock object and rule definitions are correct

1. Navigate to the following directory:

   Windows: `SIEBEL_ROOT\bin`

   UNIX: `$SIEBEL_ROOT/bin`
2. Type the following command using the parameters specified in the following table:

```
dictutl /C ODBC_DATASOURCE /U USERNAME /P PASSWORD /D TABLEOWNER /N "REPOSITORY_NAME" /A y 2>
 logfile.log
```

Flag	Parameter	Description	Required
/C	ODBC_DATASOURCE	ODBC datasource name	Yes
/U	USERNAME	User name to log in to database	Yes
/P	PASSWORD	User password to log in to database	Yes
/D	TABLEOWNER	User name of tableowner	Yes
/N	"REPOSITORY_NAME"	Name of repository for dictionary (the parameter must be bounded within double quotes)	Yes
/A	y or n	Enter the y parameter to ignore the dictionary cache. Enter n if you do not want to ignore the dictionary cache.	Yes

3. Review the `LOGFILE.log` file:
   a. Open the file in Microsoft Excel. If the file is too large for Excel to display the whole file, then break the file into two files and examine each separately.
   b. Query for the word error.
   c. If you locate errors, then write down the exact error text.
   d. Query for the word syntax.
   e. If you locate syntax errors, then write down the exact error text.
   f. To determine whether the errors must be resolved, contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.

      The following types of entries indicate that no errors were found:

      - Errors: 0
      - Syntax OK

Related Topic

Preserving Siebel Dock Objects and Visibility Rules

Verifying an Upgraded Oracle Database After a Siebel Upgrade

Environments: Development, production test, production.

Databases: Oracle only.

You can upgrade your RDBMS before or after upgrading to the current Siebel release.

If you upgrade your Oracle RDBMS after upgrading to the current Siebel release, then you must run the Siebel ddlimp utility to verify the schema layout of the upgraded database.

Note: The upgphys process might recreate descending (DESC) indexes as ascending (ASC) indexes. To recreate descending indexes, use the /9 option while running ddlimp.

This also converts the Siebel database to character-based length for char and varchar data.

To verify the upgraded Oracle Database

1. On the Siebel Server where the Siebel Database Server files are installed, navigate to the following location:

   Windows: `SIEBEL_ROOT\bin`

   UNIX: `$SIEBEL_ROOT/bin`
2. Run the following command:

```
ddlimp /u TableOwner /p TablePassword /c "ODBCDataSource" /f DBSRVR_ROOT/DatabasePlatform/schema.ddl /t
 y /i n /e n /B TableSpace /X IndexSpace /G SSE_ROLE
/R Y /l SiebelLogDir/ddlctl_verify_RDBMS.log
```

where:

- TableOwner is the Siebel table owner name, also called the schema qualifier name.
- TablePassword is the Siebel table owner password.
- ODBCDataSource is the ODBC name for connecting to the database. Enclose the name in quotes.
- DBSRVR_ROOT is the absolute path to the Siebel Database Server installation directory.
- DatabasePlatform is the Siebel Database Server directory for the Oracle Database.
- Tablespace is the Oracle tablespace name for the Siebel database.
- IndexSpace is the Oracle index space name for Siebel database.
- SiebelLogdir is the path to the directory where you want the output log placed (log output directory).

3. After the command completes, review the output log files for errors. If the log indicates there are errors, then create a service request (SR) on My Oracle Support. Alternatively, you can phone Global Customer Support directly to create a service request or get a status update on your current SR. Support phone numbers remain the same and are listed on My Oracle Support.

Related Topic

About Upgrading Your RDBMS in the Siebel Environment

Setting Oracle Database Parameters After a Siebel Upgrade

Environments: Development, production test, production.

Databases: Oracle only.

After the Siebel database upgrade is complete, set the following parameters in init.ora:

- **optimizer_index_cost_adj**. Set this parameter to 1.
- **Collecting statistics**. To optimize SQL performance, use the PL/SQL package dbms_stats to manage statistics gathering. For additional information on optimizer settings, see 781927.1 (Article ID) on My Oracle Support. This document was previously published as Siebel Technical Note 582.
- For a full list of recommended settings for your postupgrade production environment, see the chapter on configuring the RDBMS in *Siebel Installation Guide* .

18 Reviewing the Siebel User Interface

Reviewing the Siebel User Interface

This chapter provides information on reviewing your Siebel user interface following a repository merge. This chapter includes the following topics:

Troubleshooting Postmerge Siebel User Interface Problems

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

Basic Troubleshooting Guidelines

If views or screens do not display, then do the following:

- Using the Siebel Developer Web Client, verify that the user has been assigned the correct responsibilities.
- Query for the view or applet in the Administration - Personalization screen. Verify the conditional expression is correct.
- In Siebel Tools, verify that the screen view and its parents have the `Display In Site Map` property, and the `Display In Page` property set to TRUE.
- Verify that the screen view and its parents have the Viewbar Text and Menu Text properties filled in.

- If an applet does not display all the fields or controls after upgrade, then check in Siebel Tools for a Web template of the same name, but appended with `-Expanded` and specify this Web template for the applet. These templates provide additional placeholders for mapping fields and controls.

About the Postmerge Utilities Log

After the repository merge, you must run the postmerge utilities. These utilities do the following:

- Validate user interface components to verify they were migrated correctly to the new repository.
- Modify UI objects to implement new user interface features. For example, they modify form applets and multi-value group applets.
- Verify that customized UI objects are configured correctly.

The postmerge utilities log lists the actions performed by the postmerge utilities. The log contains the following types of messages:

- **STATUS.** These messages provide information on the specific things that the postmerge utilities did. No action is required, so you can ignore these messages.
- **INFO.** These messages provide information on the specific things that the postmerge utilities did. No action is required, so you can ignore these messages.
- **WARNING.** These messages provide information on UI objects that might be incorrectly configured, so must be reviewed.
- **ERROR.** These messages indicate that a problem has been found that must be corrected.

The postmerge utilities log is located here:

```
$SIEBEL_HOME\reppatch\log\reputility.log
```

For more information on troubleshooting user interface problems, see 477269.1 (Article ID) on My Oracle Support. This document was previously published as Siebel Troubleshooting Steps 19.

Related Topics

About the Siebel Postmerge Utilities

About the Siebel Repository Merge

Reviewing Siebel Grid-Based Applets

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

Reputility log section: Start Invalid Web Template Item Mapping Clean-up.

If a Prior Customer Repository applet that will be converted to grid-based layout has been customized by adding new fields or controls, then utilities puts these fields and controls at the end of the applet in the New Customer Repository. After the repository merge, you must reposition these fields and controls.

The reputility.log file lists applets that were converted by the postmerge utilities. Review the reputility log section cited at the start of this topic and revise layouts as required.

The log is located here:

```
$SIEBEL_HOME\reppatch\log\reputility.log
```

The following table lists common issues and corrective actions for grid-based applets.

Log Examples	What To Do
STATUS:: Succeed Update AWTI List Mgmt Lists Entry Applet Base Status	The utility mapped a control to a different location. **Action:** Review the modified Applet. Verify that the new location for the control works.
STATUS:: Succeed Delete AWTI List Mgmt Lists Entry Applet Base Status ESN	To avoid possible overlapping, the utilities deleted a locale record for an applet Web template item. **Action:** Review the modified Applet Web Template and re-create the locale record.
WARNING:: Grid -> Grid merge/upgrade, Skip	Only flow-based form applets are converted to the grid-based Web template. **Action:** None required.
WARNING:: Upgrade Ancestor "Account Form Applet" Not Found in New Siebel Repository, Skip	The utilities did not find the upgrade ancestor's applet Web template in the New Siebel Repository. **Action:** Review the applet for invalid Web template items.
WARNING:: Applet Web Template Not Found in New Siebel Repository, Skip	The utility is trying to find the Applet Web Template from New Siebel Repository, but it is not found. **Action:** Review the applet for invalid Web template items.
WARNING:: No List Column Found, Applet "Program Expenditure List Applet", AWT "Edit", AWTI "Type", Control "Type", Skip	No List Column found for the Applet Web Template Item. **Action:** Delete the applet Web template item.
WARNING:: No Control Found, Applet "Expense Item Entry Applet", AWT "Edit", AWTI "Site", Control "Site", Skip	No Control is found for the Applet Web Template Item. **Action:** Delete the applet Web Template.
WARNING:: Button Control, Applet "Expense Item Entry Applet", AWT "Edit", AWTI "NewQuery", Control "NewQuery", Invoke Method "NewQuery", Skip	If a control has an Invoke Method action defined on it, then the utility treats it as a button and will not remap it. **Action:** Review the applet and verify that the button control displays correctly.

Related Topics

Troubleshooting Postmerge Siebel User Interface Problems

About the Siebel Postmerge Utilities

About the Siebel Repository Merge

Reviewing Siebel User Interface Navigation

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

Reputility log section: User Interface Navigation Upgrade.

The postmerge utilities analyze the repository and verify that objects referenced in screens, views and applets are defined correctly. The reputility.log lists objects that need to be modified. Review the reputility log section cited at the start of this topic and make the necessary revisions.

The log is located here:

```
$SIEBEL_HOME\reppatch\log\reputility.log
```

The following table shows examples of common issues and corrective actions.

Log Examples	What To Do
WARNING:: Project projname is not found in this Repository	The postmerge utilities exclude certain Siebel Tools projects from UI Navigation cleanup. One of the excluded projects was not found in the repository. **Action:** None required.
WARNING:: Ignoring Screen View Record. View Definition Not Found [View: Account Briefing View]	The utility is unable to read the view definition to determine where it should display. **Action:** Remove the invalid screen view record that references the invalid view.
WARNING:: Error Writing Category Record, Ignoring Changes [Name: catname]	The utility could not update or insert a record in the Siebel database. Possible causes are that the record already exists or there is a database access problem. **Action:** Verify that a duplicate catname record does not already exist, and then check database access.
WARNING:: Error Writing Screen View Record, Ignoring Changes [View: viewname]	The utility could not create a screen view record. Possible cause is that the category does not exist. This error is often a consequence of an error updating or inserting a category record. **Action:** Verify that the screen view category exists, and then check database access.

Related Topics

Troubleshooting Postmerge Siebel User Interface Problems

About the Siebel Postmerge Utilities

About the Siebel Repository Merge

Reviewing Siebel Multi-Value Group Shuttle Applets

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

Reputility log section: Multivalue Group Shuttle Applet Upgrade.

MVG applets are shuttle-enabled by default. The postmerge utilities shuttle-enable MVG applets in the New Customer Repository. This includes MVG applets from the Prior Customer Repository that you created or customized.

MVG applets must have a specific configuration in order to be enabled as MVG shuttle applets. For information on creating and managing MVG shuttle applets, see *Configuring Siebel Business Applications* .

In the reputility.log, the utilities list MVG applets that were converted. Review the reputility log section cited at the start of this topic and resolve any problems encountered during conversion.

The log is located here:

```
$SIEBEL_HOME\reppatch\log\reputility.log
```

The following table lists common issues and corrective actions for MVG applets.

Log Entry Example	What To Do
WARNING:: [APPLET: Account Address Mvg Applet (NB)] MVG Applet is Inactive. Ignoring Applet.	This applet is inactive. **Action:** No action required.
WARNING:: [APPLET: Primary Employee Mvg Applet] [Applet Web Template: Base] Applet Web Template is Inactive. Ignoring Applet Web Template.	The applet Web template is inactive. **Action:** No action required.
WARNING:: [APPLET: FINS Application Contact Mvg Applet - ACAPS] [CONTROL METHOD INVOKED: ExecuteQuery] Has a Non Standard Control Type.	The utility is trying to map an existing Go (ExecuteQuery) button to the Edit List mode to enable Popup Inline Query. However, the Go button does not have the correct attributes. **Action:** Revise the control definition and map it to the Edit List mode.
WARNING:: [APPLET: State Model - State Mvg Applet] [CONTROL METHOD INVOKED: UndoQuery] Has a Non Standard Control Type.	The utility is trying to map an existing Cancel (UndoQuery) button to the Edit List mode to enable Popup Inline Query. However, the Cancel button defined does not have the correct attributes. **Action:** Revise the control's definition and map it to the Edit List mode.
WARNING:: [APPLET: LOY Account Address Assoc Applet]	The utility is trying to map an existing Cancel (UndoQuery) button to the Edit List mode to enable Popup Inline Query. However, the Cancel button defined is inactive.

Log Entry Example	What To Do
[CONTROL: CancelQuery] Control is Inactive. Please inspect and Reactivate.	**Action:** Redefine a Cancel button and map it to the Edit List mode.
WARNING:: [APPLET: LOY Account Address Assoc Applet] [CONTROL: ExecuteQuery] Control is Inactive. Please inspect and Reactivate.	The utility is trying to map an existing Go (ExecuteQuery) button to the Edit List mode to enable Popup Inline Query. However, the Go button defined is inactive. **Action:** Redefine a Go button and map it to the Edit List mode.
WARNING:: [APPLET: Account Address Mvg Applet] [Applet Web Template: Edit List] [Applet Web Template Item: ExecuteQuery] Applet Web Template Item is Inactive. Please inspect and Reactivate.	The utility is trying to map an existing Go/Cancel button to the Edit List mode to enable Popup Inline Query. However, the mapping already exists, but is marked inactive. **Action:** Activate the mapping (Applet Web Template Item) in the Edit List mode and test.
WARNING:: [APPLET: Account Address Mvg Applet] [CONTROL: UndoQuery] Applet Web Template Item occupying Item Id 108. Cannot Map Control UndoQuery	The utilities tried to map a Cancel button to the default location 108 in Edit List mode. However, another control is already mapped to this location. **Action:** Move the control at location 108 to another location, and then map the Go button to location 108.
WARNING:: [APPLET: Assoc Data Type Applet] Association List Applet contains both Base and Edit List. Manual review needed.	An MVG applet definition specifies both a base and edit-list Web template. The user interface standard is that when MVGs first display, they are editable. **Action:** For MVGs that users can edit, verify that the Web template specified for base mode and edit-list mode are the same. If not, change the base mode Web template so that it is the same as the edit-list template.
WARNING:: [APPLET: Activity Order Mvg Applet] [CONTROL: ExecuteQuery] Control is at an unexpected location. Expected Location is 107	The utilities tried to map a Go button to the default location 107 in Edit List mode. This is part of enabling Popup Inline Query. However, another control is already mapped to this location. **Action:** Move the control at location 107 to another location, and then map the Go button to location 107.

Related Topics

Troubleshooting Postmerge Siebel User Interface Problems

About the Siebel Postmerge Utilities

About the Siebel Repository Merge

Revising Siebel UI Rich Text Controls

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

Reputility log section: Issue 1: Rich Text Controls (RTC) That Need to Have Properties Reconfigured.

The postmerge utilities review the repository and verify that rich text controls are defined correctly. The reputility log lists the controls that need to be modified. Review the reputility log section cited at the start of this topic and make the necessary revisions.

The log is located here:

```
$SIEBEL_HOME\reppatch\log\reputility.log
```

To revise Rich Text Control definitions

1. Start Siebel Tools. Set the Object Explorer to Flat.
2. From the Views menu, choose Options, then the Object Explorer menu item.

   Under applet, verify that Applet User Prop and Control User Prop are check-marked.
3. Refer to the reputility.log and query for one of the listed applets.
4. In the Object Explorer, select Applet User Prop.

   If the user RTC Graphic Field and RTC Link Field user properties are marked Inactive, then no further action is required.
5. For the following user properties, write down the value-active RTC Graphic Field and RTC Link Field user property:
   - RTC Graphic Field. The value for RTC Graphic Field typically is Body Field Graphic.
   - RTC Link Field. The value for RTC Link Field typically is Body Field Link.
   - RTC Body Field. The value is the control name.
6. For the applet, select Control, and then Control User Prop.
7. In the Controls list, query for the value you wrote down for RTC Body Field. This is the control name.
8. For the control, select Control User Prop.
9. Define the following control user properties on the control. Assign the values you wrote down for the applet user properties:
   - RTC Graphic Field
   - RTC Link Field

Related Topics

Troubleshooting Postmerge Siebel User Interface Problems

About the Siebel Postmerge Utilities

About the Siebel Repository Merge

Reviewing New Siebel UI Aggregate Categories

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

Reputility log section: Issue 2: New Aggregate Category Records That Should Be Renamed.

If you have created new views or have modified existing views, then the postmerge utilities create new Aggregate Category records to support the new properties. The utilities name the new aggregate category records busobj_name List. For example, a new aggregate category record for the eEvents screen would be named eEvents List.

The reputility log lists the category records that were created by the postmerge utilities. Review the reputility log section cited at the start of this topic and make the following revisions as needed to the listed objects:

- Revise the Viewbar Text and Menu Text properties in all installed languages as required.
- Verify that the navigation hierarchy, including sequence numbers, is correct.

The log is located here:

```
    $SIEBEL_HOME\reppatch\log\reputility.log
```

Related Topics

Troubleshooting Postmerge Siebel User Interface Problems

About the Siebel Postmerge Utilities

About the Siebel Repository Merge

Revising Siebel Visibility Filters to Display Correctly

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

Reputility log section: Issue 3: Views that need an applet in View Web Template Item ID 1.

The postmerge utilities review the repository and verify that filters are defined correctly. The reputility log lists screens that have incorrectly defined filters. The most common problem is that none of the view Web template items has an Item Identifier of 1, which prevents the filter from displaying. Review the reputility log section cited at the start of this topic and make the necessary revisions.

The log is located here:

```
    $SIEBEL_HOME\reppatch\log\reputility.log
```

To revise visibility filters to display correctly

1. Query for one of the views listed in the reputility.log.
2. For the view, select Base, and then View Web Template Item.
3. Set the Item Identifier for the first applet in the list to 1.
4. Refine the query to display the Parent Category listed in the log.

Related Topics

Troubleshooting Postmerge Siebel User Interface Problems

About the Siebel Postmerge Utilities

About the Siebel Repository Merge

Assigning a Category and Type to Siebel Chart Views

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

Reputility log section: Issue 4: Chart Views Needing Migration to Aggregate Type.

The Chart menu item displays as one of the aggregate categories under the view tabs. To ensure that the Chart menu item is located correctly, all the relevant charts for a view must be assigned to the same Aggregate Category. Also, each chart view must be of type Aggregate View. If the chart view is not of type Aggregate View, then the chart menu item displays as a view tab.

The postmerge utilities review the repository and verify that Chart views have been defined correctly. The reputility log lists the screen views that require revision. Review the reputility log section cited at the start of this topic and make the necessary revisions.

The log is located here:

```
$SIEBEL_HOME\reppatch\log\reputility.log
```

To assign a category and type to chart views

1. Set Siebel Tools Object Explorer to Types.
2. Query for the screen.
3. In the Object Explorer, select Screen View.
4. Query for the screen views listed in the log.
5. Verify that all the views are assigned to the same category (Category Name).

   The Category Name can be null.
6. Verify that all the views are of type Aggregate View.

Related Topics

Troubleshooting Postmerge Siebel User Interface Problems

About the Siebel Postmerge Utilities

About the Siebel Repository Merge

Assigning a Category and Type to Siebel Explorer Views

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

Reputility log section: Issue 5: Explorer Views Needing Migration to Aggregate Type.

The postmerge utilities review the repository and verify that explorer views are defined correctly. The reputility.log, lists the explorer views that require revision. Review the reputility log section cited at the start of this topic and make the necessary revisions.

The log is located here:

```
$SIEBEL_HOME\reppatch\log\reputility.log
```

Assigning a New Web Template to a View

Use the following procedure to assign a new Web template to a view.

To assign a new Web template to a view

1. In Siebel Tools, query for a view name listed in the log.
2. In Object Explorer, select View Web Template.
3. In View Web Templates, change the Web Template to Tree 2.
4. Repeat for all the views listed in the log section.

Assigning a New Type to a Screen View

Use the following procedure to assign a new type to a screen view.

To assign a new type to a screen view

1. In Siebel Tools, query for the screen.
2. In Object Explorer, select Screen View.
3. For the screen views listed in the log, change the Type to Aggregate View.
4. Repeat for each screen listed in the log section.

Related Topics

Troubleshooting Postmerge Siebel User Interface Problems

About the Siebel Postmerge Utilities

About the Siebel Repository Merge

Setting Up Navigation to Inaccessible Siebel Detail Views

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

Reputility log section: Issue 6: Categories Where Parent Applets Are Missing Drilldowns to a Detail View.

The postmerge utilities verify that you can navigate from a screen to all the screen detail views listed in the parent category for the screen. If any of the screen's detail views are not accessible using normal navigation methods, then the utilities list the screen name, parent category, and the Aggregate View in the log.

In many cases, the problem is caused by a missing or incorrectly defined drilldown in a list applet in the view shown in the log. The missing drilldown prevents the user from navigating to a view containing third-level view tabs that provide access to all the detail views.

Review the reputility log section cited at the start of this topic and make the necessary revisions.

The log is located here:

```
$SIEBEL_HOME\reppatch\log\reputility.log
```

To set up navigation to inaccessible detail views

1. In Siebel Tools, navigate to the screen listed in the log. In Object Explorer, select Screen View.
2. In the Screen Views List, query for the following:
    - Set the Type value to Detail View.
    - Set the Parent Category value to the Parent Category listed in the log.
3. Start the application and navigate to the screen.
4. Try to navigate to the detail views listed in the query.

    Tip: Use the Web Layout Editor in Tools to identify a detail view containing third-level view tabs that provide navigation to all the detail views. Verify that you can navigate to this view from a drilldown in the screen.

5. When you have identified the inaccessible detail view containing third-level view tabs, review the drilldown definitions in Siebel Tools for the list applet in the screen. Define a drilldown to the detail view if one does not exist.

Related Topics

Troubleshooting Postmerge Siebel User Interface Problems

About the Siebel Postmerge Utilities

About the Siebel Repository Merge

19 Postupgrade Tasks for Siebel Business Applications

Postupgrade Tasks for Siebel Business Applications

This chapter provides tasks to be performed on Siebel Business Applications following a successful upgrade and repository merge. This chapter includes the following topics:

- *Activating License Keys*
- *Generating Siebel Reporting Relationships*
- *Setting Up Siebel Global Time Zone Support*
- *Displaying Regions in Siebel Marketing*
- *Configuring Siebel Marketing Purchase Orders for Display*
- *Upgrading Siebel Attribute Pricing*
- *Verifying Aggregate Discounts in Siebel Pricer*
- *Upgrading Inbound Siebel Workflows*

Activating License Keys

In Siebel CRM 17.0, the license keys for Siebel CRM base applications that were previously provided in seed data in the Siebel database are inactive. License keys entered by customers are unchanged. A new utility is provided for activating or deactivating the license keys that you require. You run the License Key Activation utility after installing a new Siebel database, running Incremental Repository Merge (for migration installations), or completing a full database upgrade.

You can find license key information for Siebel Business Applications at Oracle's license codes site. For the Siebel license keys, see *http://licensecodes.oracle.com/siebel.html*.

The License Key Activation utility is supported on all operating systems and databases for Siebel Business Applications.

To start the License Key Activation utility

1. On the computer where you installed Siebel Server, navigate to the following location:

   ```
   SIEBSRVR_ROOT\bin
   ```

2. Run the following program, according to your operating system:
   - Microsoft Windows: licensekeymodule.bat
   - UNIX: licensekeymodule.sh
3. Enter valid data for the following fields:
   - **Siebel Server Location.** The installation path for this Siebel Server.

   - **ODBC DSN.** The ODBC data source for the Siebel database.
   - **Table Owner.** The table owner for the Siebel database.
   - **Username.** The user name for logging into the Siebel database.
   - **Password.** The password for this user.
   - **DB Platform.** The RDBMS platform, either ORACLE, DB2UDB, DB2390, or MSSQL.
   - **Log folder.** The folder in which the log file licenseKeys.log is created. This log file shows database connection information for troubleshooting purposes, and lists all of the license keys that were activated or deactivated in each session.
4. Click Login.

   The license key activation screen appears, which lists Siebel CRM license keys.
5. For each license key module whose activation status you want to change, click the Active Flag check box to activate or deactivate this license key.
6. To apply your selections to the Siebel database, click Apply. Or, to reset any changes you have made in this screen, or since you last clicked Apply, click Reset.

   CAUTION: After you have clicked Apply, the Reset button does not reset activation settings to their original state. However, you can change the activation status and click Apply again.

Related Books

Siebel Installation Guide for the operating system you are using.

Generating Siebel Reporting Relationships

Environments: Development, production test, production.

Platforms: Windows, UNIX, IBM z/OS.

The Generate Reporting Relationships process needs to be executed after the upgrade process and whenever the denormalized hierarchy structure (S_PARTY_RPT_REL) becomes out of sync with the data in the normalized tables (S_PARTY). Tables can become out of sync in the following cases:

- After upgrading, the organizational hierarchy (even if there is only one organization) must be established to maintain appropriate visibility in the views.
- When Siebel Enterprise Integration Manager is used to import or update any of the hierarchies (positions, organizations, or access groups).

There are three visibility hierarchies: position, organization, and access groups. These hierarchies are denormalized and maintained in the table S_PARTY_RPT_REL. These denormalized hierarchies are necessary for executing visibility modes that go up or down a hierarchy. For example:

- **Manager view mode.** My Team's Accounts View displays all accounts on which managers and their subordinates are working.
- **Suborganizations view mode.** All Contacts across My Organizations View displays all contacts who associated with either my organization or any of my organization’s suborganizations.

The Generate Reporting Relationships process rebuilds the denormalized relationships in the S_PARTY_RPT_REL table so that the hierarchical view modes display the correct information. The basic operation of the function is to empty the S_PARTY_RPT_REL table and then walk through each S_PARTY record to re-create the denormalized hierarchical structures in the table. This process generates a large number of transactions for Siebel Remote users and regional nodes.

This operation is time and CPU/memory-intensive. The process might take several minutes, depending on the size and complexity of your organizational structures. Do not perform this when you are running other memory-intensive processes.

To generate reporting relationships

1. If you have an active Siebel Remote environment, then confer with an Oracle Administrator. The Administrator must arrange for the Transaction Processor component to be paused before performing this procedure.
2. In the Siebel Web Client application, navigate to Administration - Group, and then the Positions view.
3. Click the Generate Reporting Relationships button in the Position List Applet NB. Note that generating the reporting relationship might cause a large number of Siebel Remote transactions to be generated.
4. When this has completed, restart the Transaction Processor.

Setting Up Siebel Global Time Zone Support

Environments: Development, production test, production.

Platforms: Windows, UNIX, IBM z/OS.

Global deployments typically span multiple time zones. The global time zone feature converts and stores date and time data using the Universal Time Coordinated (UTC) standard. This feature enables you to track dates and times in a common format across multiple time zones.

It is strongly recommended that you operate your production environment with global time zone enabled.

For information on setting up and managing UTC, see *Siebel Global Deployment Guide* .

Displaying Regions in Siebel Marketing

Environments: Development, production test, production.

Platforms: Windows, UNIX, IBM z/OS.

Region is associated with the Region hierarchy rather than the Region LOV field. For this reason, after the upgrade the Region field for Marketing Plans, Programs, and Campaigns does not display.

To display marketing regions

1. Navigate to Administration – Location, and then Marketing Regions.
2. Create new marketing regions to correlate to the previous LOV values for the Region field.
   For more information, see *Siebel Marketing Installation and Administration Guide* .
3. Rename the Region LOV field and add it to the applets in the user interface.
   This provides backward compatibility.

Configuring Siebel Marketing Purchase Orders for Display

Environments: Development, production test, production.

At Siebel CRM 8.0, the relationship between campaign and order is changed from M:1 to M:M and is displayed in a new view, Marketing Purchase Order.

The upgrade process does not set up campaign-related purchase orders created in previous releases so that they display in the new view.

If you want to display purchase orders from previous releases, then you must run a script after the upgrade. The script changes the order type from Purchase Order to Marketing Purchase Order for campaign-related purchase orders. You can also use the script to change the order type of campaign-related sales orders to Marketing Purchase Order. This allows these purchase orders to display in the new view.

To configure marketing purchase orders

1. Run the following script against the Siebel database:

   Windows: `DBSRVR_ROOT \common\MktgPurchaseOrder.sql`

   UNIX: `DBSRVR_ROOT /common/MktgPurchaseOrder.sql`

   For campaign-related purchase orders, this script sets order type to Marketing Purchase Order. This allows these purchase orders to display in the Marketing Purchase Order view.
2. To configure campaign-related sales orders to display in the new view, open the script with a text editor, and locate the following section:

   ```
   where ORDER_TYPE_ID =
   (select ROW_ID from S_ORDER_TYPE
   where NAME = 'Purchase Order'
   ```

   Make the following change:

   ```
   where ORDER_TYPE_ID =
   (select ROW_ID from S_ORDER_TYPE
   where NAME = 'Sales Order'
   ```
3. After making the change, run the script again.

Upgrading Siebel Attribute Pricing

Environments: Development, production test, production.

Platforms: Windows, UNIX, IBM z/OS.

Note: For information on changes to Copy and Revise functionality for Quotes and Orders, see 473867.1 (Article ID). This article was previously published as Siebel Alert 1218.

At Siebel CRM 7.8, the attribute adjustments feature replaces attribute pricing in Siebel Pricer.

You must manually upgrade attribute pricing data to attribute adjustments by running a business service method. The business service method does the following:

- Upgrades attribute pricing headers to attribute adjustment headers
- Upgrades attribute pricing attributes to attribute adjustment dimensions
- Upgrades attribute pricing values to attribute adjustment dimension domains
- Upgrades attribute pricing adjustment items to attribute adjustment rules

To upgrade attribute pricing to attribute adjustments

1. Launch Siebel Sales.
2. Verify that all attribute classes have been upgraded to product classes.
3. Navigate to the Administration-Business Service screen, and then to Simulator view.
4. Create a new record.
5. Click Run to start the business method.

Related Topic

Upgrade Planning for Siebel Pricer and Order Management

Verifying Aggregate Discounts in Siebel Pricer

Environments: Development, production test, production.

Platforms: Windows, UNIX, IBM z/OS.

Note: For information on changes to Copy and Revise functionality for Quotes and Orders, see 473867.1 (Article ID). This article was previously published as Siebel Alert 1218.

At Siebel CRM 7.8, the aggregate discounts feature replaces bundle factors in Siebel Pricer. The bundle factor definitions are upgraded to aggregate discounts, and the sequencing of bundle factors are upgraded to aggregate discount sequences.

The name of the aggregate discount in Pricer will be set to bundle factor name + row ID of the record. This is because Pricer requires the aggregate discount name to be unique.

Sequencing of bundle factors within a pricing model is upgraded to aggregate discount sequencing. The name of the aggregate discount sequence is set to the pricing model name that contained the bundle factors.

The Price List and Price List Item will be stamped with the appropriate aggregate discount sequence name. In prior releases, the pricing model was specified at the Price List or, for customized products, at the Price List Line Item level. This ensured execution of the bundle factors at runtime.

At Siebel CRM 7.8, the execution of aggregate discounts at runtime requires the association of the aggregate discount sequences at the Price List or Price List Line Item level.

The upgrade process makes the following assumptions about Pricer implementations prior to Siebel CRM 7.8:

- Flowcharts were used to chain up bundle factors in the pricing model
- The bundle factor with the lowest sequence is connected to the Aggregate Start step
- Each Aggregate Start sequence contains only bundle factors and does not contain aggregate factors
- The next factor in the flowchart (when True or False) always has a larger sequence number.

If your implementation does not meet all the criteria mentioned here, then the upgrade process moves the definitions to the appropriate Pricer entities (such as aggregate discounts), but the sequences will not be correct.

In such cases, you must manually verify that aggregate discount sequences chain up the aggregate discounts as intended. Use the sequence of execution that existed prior to the upgrade.

To verify upgrade to aggregate discounts

1. Launch Siebel Sales.
2. Navigate to Administration – Pricing Aggregate Discount Sequences.
3. For each aggregate discount sequence, drill down to the detail view.
4. Locate the aggregate discount that corresponds to the first preupgrade bundle factor. Verify that it has the lowest sequence number. If not, then revise the numbers in the Sequence, Next Discount If Used, and Next Discount If Not Used columns.
5. Verify that the numbers in the Next Discount If Used and Next Discount If Not Used columns are greater than the number in the Sequence column. Also verify that they point to the expected aggregate discounts. If not, then revise the numbers in all three columns as required.

Related Topic

Upgrade Planning for Siebel Pricer and Order Management

Upgrading Inbound Siebel Workflows

Environments: Development, production test, production.

Change inbound workflows that contain a *String* type process property to pass the value into type `Binary`; otherwise, the workflow presents the following error message:

```
Output argument '<Value>' in step 'Read from File' contains data that cannot be
passed to string type property 'InputXML'. Data type: 'MEMBLOCK'; String
representation of data body: '<?xml version="1.0" encoding="UTF-8"?><?'
```

Related Topic

Upgrade Planning for Siebel Workflow Designer

20 Tuning the Siebel Upgrade Files

Tuning the Siebel Upgrade Files

This chapter provides guidelines for preparing your Siebel Business Applications data for the Siebel database upgrade. This chapter includes the following topics:

- *Starting and Stopping Siebel Upgrade Tuner*
- *Managing Parallel Threads Using Siebel Upgrade Tuner*
- *Managing Zero-Row SQL Commands Using Siebel Upgrade Tuner*
- *Transferring UNIX Files for Use by Siebel Upgrade Tuner*
- *Rolling Back Siebel Upgrade Tuner Changes*

Starting and Stopping Siebel Upgrade Tuner

Environments: Production test environment only. Does not apply to production environment.

Platforms: Windows and UNIX only.

Databases: All databases.

This topic describes how to start the Upgrade Tuner (upgtuner) from the Siebel Server installation directory on a Windows host. It also describe how to save or discard your changes when exiting Upgrade Tuner.

Use the Upgrade Tuner to improve the performance of table creation, index creation, and SQL execution during the production upgrep step in the production test environment.

Requirements

- In the production test environment, you must have completed the production upgrep step.
- You must have run the Logparse utility on the production upgrep log files. See *Summarizing Siebel Log Files Using the Logparse Utility*.
- For UNIX platforms, you must have transferred files to a Windows host on which a Siebel Server is installed. See *Transferring UNIX Files for Use by Siebel Upgrade Tuner*.

Starting Upgrade Tuner

Use this procedure to start Upgrade Tuner.

To start Upgrade Tuner

- Enter the following command:

```
SIEBEL_ROOT\bin\upgtuner /c LOGPARSE_SUMMARY_LOCATION
```

LOGPARSE_SUMMARY_LOCATION is the location of the summary.xml file generated by the Logparse utility for the production upgrep. For example:

```
upgtuner /c SIEBEL_ROOT\log\upgrep_prod_77\summary.xml
```

The Upgrade Tuner Process Information page appears.

Alternative: Start Upgrade Tuner without the `/c` option. Click OK at the error pop-up. When Upgrade Tuner displays, click Browse and navigate to the summary.xml file.

Saving Your Changes and Exiting Upgrade Tuner

Use this procedure to save your changes and exit Upgrade Tuner.

To save your changes and exit Upgrade Tuner

1. Evaluate the changes you have made and revise them as needed.

   Upgrade Tuner does not revise the upgrade files until you exit.
2. Click Save and then Exit.
3. Click Yes in the pop-up window that asks you to confirm if you want to save and exit.

   Upgrade Tuner applies your changes to the upgrade files and exits.

Discarding Your Session Changes and Exiting Upgrade Tuner

Use this procedure to discard the changes you have made in the current session and exit Upgrade Tuner. No changes are made to upgrade files.

Parallelize Table or Index Creation Pages

Changes are discarded and do not display the next time you start Upgrade Tuner, if:

- You moved tables or indexes between threads, these changes are discarded.
- You created new threads, the threads are discarded.

Deactivate 0-Row SQLs

Changes are discarded and do not display the next time you start Upgrade Tuner, if:

- You deactivated a command, this change is discarded. The next time you start Upgrade Tuner, no check mark displays in the Inactive column. The command remains active in the SQL file.
- You activated a command, this change is discarded. The next time you start Upgrade Tuner, a check mark displays in the Inactive column, and the command remains inactive in the SQL file.

To discard your changes and exit Upgrade Tuner

1. Click Cancel.
2. Click Yes in the pop-up window that asks if you to confirm you want to discard your changes and exit.

Related Topic

About Tuning Siebel Production Upgrade Files

Managing Parallel Threads Using Siebel Upgrade Tuner

Environments: Production test environment only. Does not apply to production environment.

Platforms: Windows and UNIX only.

Databases: Oracle only.

Upgrade Tuner allows you to create, edit, and delete parallel thread for table and index creation. This improves upgrade performance by reducing the amount of time required to complete table and index creation.

You create, edit, and delete threads in the Parallelize Table Creation page and the Parallelize Index Creation page. These two pages have the same layout.

Requirements

- In the production test environment, you must have completed the production upgrep step.
- You must have run the Logparse utility on the production upgrep log files. See *Summarizing Siebel Log Files Using the Logparse Utility*.
- For UNIX platforms, you must have transferred files to a Windows host on which a Siebel Server is installed. See *Transferring UNIX Files for Use by Siebel Upgrade Tuner*.
- Start Upgrade Tuner. See *Starting and Stopping Siebel Upgrade Tuner*.

Displaying Threads

You can view and sort the contents of threads or view items sorted across threads:

- **Default sort.** The default sort is all the items in the Serial Thread sorted from highest to lowest cost. The default sort displays when you start Upgrade Tuner, add a thread, or remove a thread. To reverse the sort order, click the Serial Thread column head.
- **Contents of a thread.** Click the column head for that thread. To reverse the sort-order, click the column head again.
- **All items in all threads sorted by cost.** Click the Cost per Table column head. To reverse the sort-order, click the column head again.

Creating Parallel Threads

Use this procedure to create a parallel thread for table creation or index creation. Upgrade Tuner automatically names threads Parallel Thread 1, Parallel Thread 2, and so on. You cannot edit thread names.

You start creating parallel threads by creating Parallel Thread 1 and Parallel Thread 2 together. You must assign at least one table or index to each of these threads.

All threads you create must contain at least one table or index.

Creating Parallel Thread 1 and Parallel Thread 2

Use the following procedure to create Parallel Thread 1 and Parallel Thread 2.

To create Parallel Thread 1 and Parallel Thread 2

1. In the Serial Thread, select a table or index, and move it to the right using the forward arrow key on the keyboard.

   Upgrade Tuner creates Parallel Thread 1 and Parallel Thread 2. It then assigns the table or index to Parallel Thread 1.
2. Move at least one table or index to Parallel Thread 2, to populate both threads.

Creating Additional Parallel Threads

Use the following procedure to create additional parallel threads.

To create additional parallel threads

- Select a table or index and move it to the right using the forward arrow key on the keyboard.

  When you move the table or index to the highest-numbered thread and click the arrow key again, Upgrade Tuner creates a new thread and places the table or index in the new thread.

Tip: Another way to create a new thread is to right-click in a row. In the drop-down menu, select the last thread listed.

Moving Items Between Threads

Use the arrow keys (forward and back) to move tables or indexes between threads, including the Serial Thread.

Note: You cannot save and exit if any thread is empty.

Deleting a Thread

Use this procedure to delete an existing thread. You cannot delete the Serial Thread.

You must delete Parallel Thread 1 and Parallel Thread 2 together. You must delete all other threads before deleting Parallel Thread 1 and Parallel Thread 2.

Deleting a Thread Other Than Parallel Thread 1 or Parallel Thread 2

Use the following procedure to delete a thread other than Parallel Thread 1 or Parallel Thread 2.

To delete a thread other than Parallel Thread 1 or Parallel Thread 2

1. Click the column head for the desired thread.

   This action sorts the list so that all items in that thread appear at the start of the list.
2. Right-click in the column head, and choose Move all items to the serial thread from the pop-up menu.
3. Right-click in the column head, and choose Remove thread from the pop-up menu.

   Upgrade Tuner deletes the thread and renames all higher-numbered threads.

Deleting Parallel Thread 1 and Parallel Thread 2

Use the following procedure to delete Parallel Thread 1 and Parallel Thread 2.

To delete Parallel Thread 1 and Parallel Thread 2

1. Right-click the column head for Parallel Thread 2, and choose Move all items to the serial thread from the pop-up menu.
2. Right-click the column head for Parallel Thread 1, and choose Move all items to the serial thread from the pop-up menu.
3. Right-click in the column head for Parallel Thread 2, and choose Remove thread from the pop-up menu.

   Upgrade Tuner deletes Parallel Thread 1 and Parallel Thread 2.

Evaluating Upgrade Performance Improvement

To evaluate production upgrep performance improvement, use the two fields at the start of the page:

- **Total cost of sequential table (or index) creation.** Displays the time to create tables or indexes when no parallel threads are used.
- **Total cost of parallelized table (or index) creation.** Displays the time to complete the upgrade using the parallel threads you have created. The time is computed by adding the Serial Thread time and the longest-running parallel thread time.

The difference between the sequential creation time and the parallelized creation time is an estimate of the reduction in upgrade time from using parallel threads.

You can reduce upgrade time further by performing the following actions:

- Move additional items from the Serial Thread to a parallel thread
- Move items from the longest-running parallel thread to other threads or a new thread

The goal is to reduce both the Serial Thread time and longest-running parallel thread time to a minimum. Because each new parallel thread requires additional memory and CPU cycles, you might need to experiment with the number of parallel threads to optimize upgrade performance.

Related Topic

About Tuning Siebel Production Upgrade Files

Managing Zero-Row SQL Commands Using Siebel Upgrade Tuner

Environments: Production test environment only. Does not apply to production environment.

Platforms: Windows and UNIX only.

Databases: All databases.

The upgrade scripts support all the tables in the Siebel data model. This support means the tables might contain SQL commands that run against tables that are not included in your Siebel database, that are empty, or do not contain data that applies to a specific SQL command. By inactivating such SQL commands, you can reduce the time required to perform the production upgrep.

The Deactivate 0-Row SQLs page displays a list of SQL files that contain commands that returned zero rows. This means the command does not affect any data and does not change the database schema. The screen displays only upgrade commands executed natively by the RDBMS. The screen does not display SQL commands executed using odbcsql.

The SQL files are located in `DBSRVR_ROOT\DBPLATFORM\upgrade\VERSION` , for example, `db2\upgrade\V8_x`.

When you select a file, the command that returned zero rows displays in the lower half of the screen. You can then either deactivate or activate the command. You cannot edit the displayed command.

When you deactivate a command and save your changes, Upgrade Tuner opens the SQL file containing the command and inserts `(Execute=No)` in the command. When you activate a command, Upgrade Tuner removes `(Execute=No)` from the command.

Requirements

- In the production test environment, you must have completed the production upgrep step.
- You must have run the Logparse utility on the production upgrep log files. See *Summarizing Siebel Log Files Using the Logparse Utility*
- For UNIX platforms, you must have transferred files to a Windows host on which a Siebel Server is installed. See *Transferring UNIX Files for Use by Siebel Upgrade Tuner*
- Analyze your customizations and the nature of application data. Verify that you understand the role of any new tables you have added.
- Start Upgrade Tuner. See *Starting and Stopping Siebel Upgrade Tuner*

Displaying Zero-Row SQLs

You can view and sort zero-row SQLs in several ways:

- **Default sort.** The default sort order is the order in which the zero-row commands appear in the driver files. Any inactivated SQL commands, including those inactivated in previous sessions appear at the end of the list. The default sort order displays when you start Upgrade Tuner.
- **Display items sorted by cost.** To sort commands from longest-running time to shortest, click the Net Cost column head. To reverse the sort order, click Net Cost again. Commands inactivated prior to this session appear at the end of the list.
- **Display commands activated or deactivated in the current session.** Click the Inactive column head. Items display at the beginning of the list. The word Changed displays in the Inactive column for these items. Items that have been deactivated display check marks. Items that have been activated do not.
- **Display commands inactivated in previous sessions.** Click the Net Cost column head and scroll to the end of the list. Inactivated commands do not have a check mark in the Inactive column and do not display the word Changed.
- **Display commands activated in previous sessions.** The display of SQL commands does not provide a way to identify commands activated in a previous Upgrade Tuner session. When you activate a command, write down its SQL file name and SQL tag number so you can locate the command in future sessions.
- **Display all the zero-row SQL commands in a file.** Click the SQL File column head. This action sorts the file names alphabetically. To reverse the sort order, click the column head again.

Deactivating Zero-Row SQL Commands

Use this procedure to deactivate SQL commands that do not affect any data.

To deactivate zero-row SQL commands

1. Click the Deactivate 0-Row SQLs tab in Upgrade Tuner.

   The Deactivate 0-Row SQLs screen appears.
2. Click the Net Cost column head.

   This sorts the entries so that the longest running SQL commands appear first. If they do not, click the column head again. Commands deactivated in previous Upgrade Tuner sessions display at the end of the list.
3. Click in a row to display a command that returned zero rows.
4. Carefully evaluate whether you need this command for your upgrade.
5. Write down the net cost of the command.

   You can use a spreadsheet to keep track of net cost changes, if you prefer.
6. To deactivate the command, click in the check box in the Inactive column.

   The following occurs:

   - A check mark displays indicating the command is inactive.
   - The word Changed appears next to the check mark to indicate the change was made in this session.
   - The time displayed in the Net Cost column changes to Not applicable.

- When you save and exit, Upgrade Tuner inactivates the command in the SQL file.
- The next time you start Upgrade Tuner, a check mark displays in the Inactive column for the command, but the word Changed does not.

Activating Zero-Row SQL Commands

Use this procedure to activate SQL statements that do not affect any data.

To activate zero-row SQL commands

1. Click the Net Cost column head, and then scroll to the end of the list.

   This sorts commands by running time. Inactive commands have a running time of Not applicable and always appear at the end of the list.
2. Click in a row to display a command that returned zero rows.
3. Carefully evaluate whether you need this command for your upgrade.
4. To activate the command, click in the check box in the Inactive column.

   The following occurs:

   - The check mark disappears from the check box, indicating the command is active.
   - The word Changed appears next to the check box to indicate the change was made in this session.
   - The time displayed in the Net Cost column remains Not applicable.
   - When you save and exit, Upgrade Tuner activates the command in the SQL file.
   - The next time you start Upgrade Tuner, Not applicable is replaced by the running time for the command, and the word Changed does not appear.
5. Write down the SQL file name and SQL tag number for the command.
6. The next time you run Upgrade Tuner, locate the command and write down its net cost.

   You can use a spreadsheet to keep track of net cost changes, if you prefer.

Evaluating Upgrade Performance Improvement

To evaluate production upgrep performance improvement, add together the net cost of all the zero-row SQLs you deactivated. Then subtract the net cost of the zero-row SQLs you activated.

The final sum is an estimate of how much you have reduced the time required for the next production upgrep.

Related Topic

About Tuning Siebel Production Upgrade Files

Transferring UNIX Files for Use by Siebel Upgrade Tuner

Environments: Production test environment only. Does not apply to production environment.

Platforms: UNIX only.

Upgrade Tuner is part of the Siebel Server and runs only on Windows. To tune production upgrade files on a UNIX computer you must do the following:

- Transfer upgrade files needed by Upgrade Tuner from the UNIX host to a Windows host on which a Siebel Server is installed.
- Run Upgrade Tuner on the Windows host using the UNIX files as input.
- Transfer the modified upgrade files from the Windows host back to the UNIX host.

Scripts generated by the Logparse utility during the production upgrep on the UNIX host simplify the file transfer process:

- **upgtuner_ftp_get.txt.** This script moves upgrade files from a UNIX host to a target directory on the Windows host.
- **upgtuner_ftp_put.txt.** This script moves the upgrade files from the Windows host to a target directory on the UNIX host.

Requirements for the UNIX Host

- In the production test environment, you must have completed the production upgrep step.
- You must have run the Logparse utility on the production upgrep log files. See *Summarizing Siebel Log Files Using the Logparse Utility*.

Requirements for the Windows Host

- You must have installed a Siebel Server. You do not have to install the Siebel Database Server.
- The Windows host can be a Siebel Server on which have performed upgrades. Tuning upgrade files from a UNIX host does not interfere with upgrade files already on the Windows-based Siebel Server.
- To run the `upgtuner_ftp_put.txt` script, you must be able to FTP from the Windows host to the UNIX host.

The following procedures use FTP to transfer files. If FTP is not available, you can use other methods for transferring files.

Transferring Files from the UNIX Host to the Windows Host

To run Upgrade Tuner on UNIX upgrade files, you must first transfer the files to a Windows host.

To transfer files from the UNIX host to the Windows host

1. **Windows host.** Create a target directory for the UNIX upgrade files, and share the directory.
2. **UNIX host.** Copy the following scripts using FTP to the Windows computer target directory:

- `upgtuner_ftp_get.txt`
- `upgtuner_ftp_put.txt`

The files are located in `$SIEBEL_ROOT/bin`.

3. **Windows host.** In both scripts, replace placeholder parameters with actual values, as described in the following table.

Placeholder	Value
&HostIP	This value is the IP address of UNIX computer.
&Username	This value is the user name used to open an FTP session with the UNIX computer (for example sadmin).
&WindowsTempDir	This value is the full path of the target directory on the Windows computer. The target directory does not have to be within the Siebel Server installation. Avoid using a target directory that already contains upgrade files.

4. **Windows host.** Use FTP and `upgtuner_ftp_get.txt` to move the files shown in the following table from the UNIX host to the target directory on the Windows host.

File	Location on the UNIX Host
`summary.xml`	`$SIEBEL_ROOT/log/upgrep_prod_VERSION` For example, `$SIEBEL_ROOT/log/upgrep_prod_8x/summary.xml`.
`master_upgrep_prod_VERSION.ucf`	`$SIEBEL_ROOT/bin` For example,`$SIEBEL_ROOT/bin/master_upgrep_prod_8x.ucf`.
`schema*.ddl`	`DBSRVR_ROOT/DBPLATFORM` For example, `DBSRVR_ROOT/Oracle/schema.ddl, schema_t1.ddl, schema_t2.ddl`.
`driver_upgrep_prod_VERSION.ucf`	`DBSRVR_ROOT/DBPLATFORM/upgrade/VERSION` For upgrades from Siebel CRM version 8.1.1.10 on Oracle Database, an example would be `DBSRVR_ROOT/Oracle/upgrade/v8_1_1_10/driver_upgrep_prod_v81110.ucf`.
`*.sql`	`DBSRVR_ROOT/DBPLATFORM/upgrade/VERSION`

File	Location on the UNIX Host
	For example, for upgrades from Siebel CRM version 8.1.1.10 on Oracle Database, an example would be `DBSRVR_ROOT /Oracle/upgrade/v8_1_1_10/preschm.sql`.

5. **Windows host.** Navigate to the target directory containing the UNIX upgrade files, and open the summary.xml file in a text editor.
6. **Windows host.** Near the beginning of the file, locate the element `<SIEBEL_ROOT>`, and edit the value to be the absolute path to the target directory containing the UNIX files that you copied to the Windows host.
7. **Windows host.** Save the file, and exit.
8. **Windows host.** Start Upgrade Tuner, and tune the UNIX upgrade files.

   Specify the target directory containing the UNIX upgrade files. The summary.xml file contains a flag that tells Upgrade Tuner to look for all the upgrade files in the target directory. You do not have to move the files.

Transferring Files from the Windows Host to the UNIX Host

After you have tuned the UNIX upgrade files, transfer them back to the UNIX host.

To transfer files from the Windows host to the UNIX host

1. **UNIX host.** Create a target directory for the UNIX upgrade files that will be transferred from the Windows host.

   Alternative: Use the FTP upload directory for the UNIX host.
2. **Windows host.** Use FTP to copy the UNIX upgrade files from the target directory to the UNIX host.
3. **UNIX host.** Move the upgrade files to their proper locations.

   The path for `<SIEBEL_ROOT>` in the summary.xml file is used for the Windows host and thus is incorrect for the UNIX host. The next time you run Logparse, it will overwrite summary.xml and include the path for SIEBEL_ROOT on the UNIX host computer.

Related Topic

About Tuning Siebel Production Upgrade Files

Rolling Back Siebel Upgrade Tuner Changes

Environments: Production test environment only. Does not apply to production environment.

Platforms: Windows and UNIX only.

Databases: All databases.

Use these procedures to discard the changes you saved from the most recent Upgrade Tuner session. You do this by rolling back the upgrade files to a previous Upgrade Tuner session.

This roll-back process is particularly useful for deployments on UNIX. You can roll back upgrade files to a previous version on the UNIX host. You do not have to transfer the files to a Windows host and rerun Upgrade Tuner.

Upgrade File Versions

Before Upgrade Tuner saves changes to the upgrade files, it does the following:

- If this your first Upgrade Tuner session, Upgrade Tuner saves the current driver and SQL files to .orig, for example `driver_upgrep_prod_8x.ucf.orig`.
- If this is the second session or all later sessions, Upgrade Tuner saves the current driver and SQL files to .old, for example `driver_upgrep_prod_8x.ucf.old`.

To roll back, you replace the upgrade file with the .old or .orig version.

Guidelines for Rolling Back Upgrade Files

Use the following guidelines to roll back upgrade files:

- To roll back to the previous Upgrade Tuner session, replace the driver or SQL file with the .old version.
- To roll back to the original version of the file, before any Upgrade Tuner modifications, replace the driver or SQL file with the .orig version.
- You can roll back the driver and SQL files separately. For example, you can roll-back to the original driver file while retaining the most recent changes to SQL files.
- You do not have to roll back all SQL files at once. For example, you can roll back some SQL files while retaining the most current version of others.
- You do not have to roll back all the commands in an SQL file. You can edit the file and activate or deactivate as many commands as desired.
- Manually editing the `driver_upgrep_prod_VERSION.ucf` file is not recommended.
- If you roll back to a session with fewer threads, then you do not have to delete any `schema_t#.ddl` or `schema_i#.ddl` thread-files. The deletion is unnecessary because Upgrade Tuner removes from the driver file the steps that execute the deleted threads.

The following procedures use `driver_upgrep_prod_8x.ucf` and `pret.sql` as examples.

Rolling Back to the Previous Session

Use this procedure to discard your most-recent session and roll back to the previous session.

To roll back to the previous session

1. Save a copy of `driver_upgrep_prod_8x` and `pret.sql` to new names.
2. Copy `driver_upgrep_prod_8x.old` to `driver_upgrep_prod_8x` .
3. Copy `pret.sql.old` to `pret.sql`.
4. Restart Upgrade Tuner.

Rolling Back to the Original Upgrade Files

Use this procedure to discard all Upgrade Tuner changes and roll back to the original upgrade files.

To roll back to the original upgrade files

1. Save a copy of `driver_upgrep_prod_8x` and `pret.sql`.
2. Copy `driver_upgrep_prod_8x.orig` to `driver_upgrep_prod_8x` .
3. Copy `pret.sql.org` to `pret.sql`.

Activating or Deactivating SQL Commands Manually

Use this procedure to activate or deactivate individual zero-row SQL commands by editing the SQL file.

To activate or deactivate a command by editing an SQL file

1. Save a copy of the SQL file to a new name.
2. Open the .sql file (not the copy) and locate the desired SQL command.

   Commands begin `Run_SQL_#`, for example `Run_SQL_100`.
3. Edit the command as follows:
   - To activate the command, delete the element (Execute=N)
   - To deactivate a command, add the element (Execute=N)

     Insert the element on a line by itself after `Run_SQL_# =`.
4. Save the file.

Related Topic

About Tuning Siebel Production Upgrade Files

21 Migration Planning Using Siebel Migration

Migration Planning Using Siebel Migration

This chapter provides information on planning your data migration (for a Dev to Test to Production) with Siebel Migration. It includes the following topics:

- *About Migrating with Siebel Migration*
- *Roadmap for Planning a Migration with Siebel Migration*
- *About Migration Process Orchestration During the Siebel Migration Process*
- *About the Process Flow for Migration Resources*
- *About the Siebel Migration Log Files*
- *About REST API Used for Migration Discovery and Execution*

Note: The Siebel Migration Application is supported in Siebel CRM 17.0 and later releases. The Siebel Migration Application was previously known as the Repository Migration Utility (dev2prod, which is no longer supported).

About Migrating with Siebel Migration

Environments: Development, production test, production.

Platforms: Windows, UNIX, IBM z/OS.

Siebel Migration is a Web-based tool for end-to-end repository and data migration. Migrating repository or data using Siebel Migration is not the same as performing a database upgrade where you migrate your custom repository and schema from one release of Siebel CRM to a higher release level. Migrating repository or data using Siebel Migration is a tool that allows you to replicate the setup (including repository, Runtime Repository, application Workspace data, application data, application interface Web artifacts, and file system artifacts) that exists on one environment (known as the source) to another environment (known as the target).

- **Source.** The source environment is the Design Time Repository (DR), also known as the development environment.
- **Target.** The target environment is the Runtime Repository (RR), also known as the test environment (system integration testing or user acceptance testing) or production (live environment) or pre-production environment that internal and external users access.

Note: Siebel Migration doesn't support:

- Migration of repository or Workspace data from RR (non-development) environments to other RR environments. Migrating from Test to Production or UAT to Production is not permitted.
- Cross database vendor platform migrations such as Oracle to MSSQL, DB2 to Oracle, and so on.

CAUTION: It is recommended that you back up your target database before starting to migrate repository or data using Siebel Migration.

Siebel Migration uses the RESTful services to export the data on the source environment, transfer the exported data to the target environment, and then import the data to the target environment. Siebel Migration orchestrates all the resources chosen in the migration plan that is being executed.

Siebel Migration is capable of synchronous and asynchronous migration of database artifacts. For example:

- **Synchronous migration.** You can use Siebel Migration for the synchronous migration of your repository, Runtime Repository, application Workspace data, application data, application interface Web artifacts and file system artifacts from a source environment to a target environment.
- **Asynchronous migration.** You can also use Siebel Migration for asynchronous migration plans, where all activities that must be completed on the source environment can be done independently of all the migration activities to be completed on the target environment. This is beneficial if you have firewall restrictions or situations where source environments and target environments are managed by different teams. Creating asynchronous migration plans on the source and target environments allows customers to separate responsibilities by assigning a team to activities on the source environment and assigning another team to activities on the target environment.

Siebel Migration provides the following tools to prepare your data for migration:

- Database Utilities.
- Application Deployment Manager (ADM).

Using Siebel Migration, you can to do the following:

1. **Add connections to your migration.** For more information about creating connections, see *Creating a Connection*.
2. **Create migration plans.** For more information about creating migration plans, see *Creating a Migration Plan*. Migration plans are segregated into two separate plans:
    - **Source Only Environment migration plan.** This migration plan exports all the required artifacts based on the resources selected as a package. If you create a migration plan for the source only environment, the plan can be used for export only.
    - **Destination Only Environment migration plan.** This migration plan is on the target environment that imports all the required artifacts. This migration plan is based on the same resources selected for the export migration plan on the source environment. If you create a migration plan for a destination only target environment, then the migration plan can be used for import only.
3. **View the historical data of the migration execution and log history for migration tasks.** For more information about viewing historical data of the migration execution, see *Viewing Migration History and Log Files*.
4. **Execute migration plans.** The migration plan execution list includes the Package Filename field. This field is populated with the package filename provided by the user. Depending on the action selected and the resources or services selected when you execute the migration plan, you will be prompted to enter additional information. For more information about executing migration plans, see *Executing a Siebel Migration Plan*.
    - **Export Only.** When you execute an Export Only Migration Plan on a source, Siebel Migration exports the data for all the selected resources and creates a package ZIP file. The package ZIP file will be created in the Migration Package Location provided the location is set in the migration profile, otherwise the package ZIP file will be created in the `<File System>\migration` folder on the source environment.

      You use this package to import data to a destination environment. You must take the package ZIP file from the source environment and place it on the destination environment (in the Migration Package Location or `<File System>\migration` folder). Once the exported package is placed on the destination environment, you can run the migration plan as Destination Only and the Siebel Migration Application imports the package file.

> **Note:** If the Siebel Migration Application that connects to the source environment and the Siebel Migration Application that connects to the target environment are both using the same Migration Package Location, then you do not need to copy the package ZIP file in the `<File System>\migration` folder on the destination environment.

> **Note:** The Siebel service owner account must have read-write access to Siebel File System and the Migration Package Location. For more information about creating the Siebel service owner account, see *Siebel Installation Guide*.

When you execute an Export Only Migration Plan, Siebel Migration creates a manifest file and exports the data for the selected resources. The manifest file contains the list of resources that were exported as part of this execution and the watermark filename.

- **Import Only.** When you execute an Import Only Migration Plan using the package filename, Siebel Migration verifies that the resources selected in the migration plan matches the resources written in the manifest file. Siebel Migration also verifies that the watermark present in the watermark file and the manifest file matches the watermark on the connection where the user is importing. The execution proceeds only if both the resource and watermark matches.

> **Note:** Watermarks are only matched for Incremental Runtime Repository and File Prepare and Deploy resources.

5. **Use REST APIs.** You can also use REST APIs to interact with the Siebel Migration Application. You must run the repository upgrade before you use REST APIs with the Siebel Migration Application. For more information about using REST APIs with the Siebel Migration Application, see *Using REST API with the Siebel Migration Application*.

The following chapters deal with Siebel Migration:

- *Migration Planning Using Siebel Migration*
- *Data Preparation for Siebel Migration*
- *Data Migration Using Siebel Migration*

Roadmap for Planning a Migration with Siebel Migration

Environments: Development, production test, production.

Platforms: Windows and UNIX only.

This topic provides an overview of the recommended guidelines for planning and managing the data migration process.

Use the following steps to help plan your migration.

1. **Install Siebel Migration.** Siebel Migration is installed with Siebel Application Interface as part of the Siebel Enterprise Server software installation. For more information about installing Siebel Application Interface, see *Siebel Installation Guide for Microsoft Windows* .

2. **Configure Siebel Migration with Siebel Management Console.** Siebel Management Console is installed with Siebel Application Interface as part of the Siebel Enterprise Server software installation. Configuring Siebel Migration consists of the following tasks:
   a. **Create the Siebel Migration Profile.** The Siebel Migration profile is created with the Siebel Management Console. For more information about creating the Siebel Migration profile, see *Siebel Installation Guide for Microsoft Windows* .

      Optionally, you can enter a Migration Package Location when you create a Siebel Migration profile. You must give a network file share path. If the Migration Package Location is configured in the migration profile in Siebel Management Console, the Export, Import and Generate Watermark actions use the Migration Package Location instead of using the migration folder in the file system. If the Migration Package Location field is provided, Siebel Migration copies the exported package file in the Migration Package Location and imports the specified package file from this Migration Package Location instead of the migration folder.

      > **Note:** The process running the Siebel Object Manager must have read-write access to the network file share path for the Migration Package Location.

   b. **Configure Siebel Migration.** Siebel Management Console comes with a pre-seed profile. You can either edit the existing pre-seed profile, create a new profile, or create a new profile by cloning the pre-seed profile. For more information about configuring Siebel Migration, see *Siebel Installation Guide for Microsoft Windows* .
   c. **Deploy the Siebel Migration Profile.** The Siebel Migration profile is deployed with the Siebel Management Console. For more information about deploying the Siebel Migration profile, see *Siebel Installation Guide for Microsoft Windows* .
3. **Configure Authentication for Siebel Migration.** Siebel Migration supports Basic and SSO authentication. You select the authentication type when creating the Siebel Migration profile in Siebel Management Console. For more information, see *Siebel Installation Guide for Microsoft Windows* .
   - **Basic Authentication:** Siebel REST services authenticate Siebel Migration users. Basic authentication internally uses the AuthenticateUser method from the Authentication Service For Migration RESTful Service to ensure whether the user has a permission to access the application.
   - **SSO Authentication:** The User is authenticated by the SSO server. Once the user is successfully authenticated, the request is forwarded to Siebel Migration. Siebel Migration uses the AuthenticateUser method from Authentication Service For Migration RESTful services to ensure whether a user has a permission to access Siebel Migration.
   - **Migration REST Endpoint for Authentication and Data Store:** This setting specifies the REST endpoint URL for the Siebel Migration Application. The REST endpoint URL is used to authenticate the Siebel Migration user and to store the Siebel Migration Application data. The format of the URL is as follows where AI_HOST is the Application Interface host name and AI_PORT is the port number: `https://AI_HOME:AI_PORT/siebel/v1.0`. You can optionally replace the keyword `siebel` as described in Customizing URLs for Siebel CRM Applications in *Siebel Installation Guide for Microsoft Windows* .

   Siebel Migration invokes the Authentication Service For Migration business service to authenticate a user. You must configure the Authentication Service for Migration business service to restrict the access to the Siebel Migration Application by adding responsibilities to the AuthenticateUser method. For more information about configuring responsibilities and access control for business services, see *Siebel Security Guide* .
4. **Configure REST Inbound in Siebel Management Console.** Siebel Migration uses RESTful service. REST Authentication and REST Inbound Defaults are configured in Siebel Management Console as part of the Siebel Application Interface Profile. For more information about configuring REST Inbound in Siebel Management Console, see *Siebel REST API Guide* and *Siebel Installation Guide for Microsoft Windows* .

5. **Setting Up the Siebel Environments.** The source and target environments must have several Siebel Server components enabled. For more information about Siebel Server components, see *Siebel System Administration Guide* .

   The source Siebel environment must have the following Siebel Server components enabled:

   - Workflow Management Component Group - Workflow Process Manager
   - Enterprise Application Integration Component Group - EAI Object Manager
   - Siebel Remote Component Group - Synchronization Manager

   The target Siebel environment must have the following Siebel Server components enabled.

   - Workflow Management Component Group - Workflow Process Manager
   - Enterprise Application Integration Component Group - EAI Object Manager

   In order to use the Siebel Migration Application in a Synchronous mode, the Migration user (such as SADMIN) must have the same username and password for both the Source and Target environments. For use in Asynchronous mode, the username and password can be different in Source and Target environments.
6. **Set the LDR_CNTRL environment variable for AIX environments.** For AIX environments, you must set the value of the LDR_CNTRL environment variable to the following:

   ```
   LDR_CNTRL=LOADPUBLIC@MAXDATA=0x60000000
   ```

   For more information about the LDR_CNTRL environment value, see *Siebel Performance Tuning Guide* .
7. **Prepare the migration data.** Siebel Migration exposes Database Utilities that you can use to prepare your data for migration. For more information about preparing data for migration, see *Process of Preparing Siebel Application Data for Migration*.

   a. **Create Application Deployment Manager projects.** Use Siebel Application Deployment Manager to create Application Deployment projects. For more information about creating Application Deployment Manager projects, see *Siebel Application Deployment Manager Guide* .

   > **Note:** To export the ADM Package on the source environment, you must create a Deployment Project with the Export to File flag set to TRUE in the Deployment Projects view in the Application Deployment Manager screen. However, the Export to File flag must not be set to TRUE for Export and Import together. In the Source environment, to export the ADM Project into a file, the EAIFileTransportFolders parameter should have the `<Siebel File System>\migration` folder configured. Otherwise, the export file fails due to a write permission issue. For more information about enabling write access for the EAI File Transport, see *Transports and Interfaces: Siebel Enterprise Application Integration* . To import the exported package on the Destination environment, the Deployment Project Name must be same as the Source Deployment Project Name or the import will fail.

   b. **Use Application Deployment Manager to transform data.** Use Application Deployment Manager to create data maps to transform your migration data. For more information about transforming data with Application Deployment Manager, see *Process of Transforming Data with Siebel Application Deployment Manager* .

   c. **Customize Migration Process Orchestration.** You can add new migration resources to the `ResourceSequence.txt`. The Siebel Migration Application reads through this file during the execution process and executes the services in a sequential order. You can customize the sequence of the migration process by modifying the ResourceSequence.txt file. For more information about customizing the migration process orchestration, see *About Migration Process Orchestration During the Siebel Migration Process*.

8. **Use Siebel Migration.** Use Siebel Migration to add connections to a migration, create migration plans, execute migration plans, and review migration history. For more information about using Siebel Migration, see *Data Migration Using Siebel Migration*.
9. **Review Migration Log Files.** Use Siebel Migration to review migration log files. For more information about migration log files, see *About the Siebel Migration Log Files*.

About Migration Process Orchestration During the Siebel Migration Process

Environments: Development, production test, production.

Platforms: Windows, UNIX, IBM z/OS.

During the migration process, the Siebel Migration Application executes a set of business service methods for each resource. The execution sequence of these service methods is defined by an external sequence text file, the ResourceSequence.txt file. The ResourceSequence.txt file lists the names of the Business Services for each resource and the supported methods for Export, Import, and Status. The ResourceSequence.txt file defines the order of migration execution.

The Siebel Migration Application reads through this file during the execution process and executes the services in a sequential order.

The migration executes a resource only if the resource is present in the ResourceSequence.txt file. If the resource is not defined in the ResourceSequence.txt file, then the resource will not appear in the Siebel Migration and will not be executed.

New migration resources can be added to the hen t file in the required execution order. New business services should adhere to the following standards:

1. The business service must support methods for Import, Export, and GetStatus.
2. Any extra methods or extra Input or Output to these methods are exceptions and requires handling in the Siebel Migration code before inclusion in the sequence file.
3. If the methods are Sync methods that do not support GetStatus, then the execution will proceed based on the HTTP response and the output parameters will not be parsed. If parsing is required for the output parameters, then it should be handled in the Siebel Migration code.
4. Oracle recommends that you do not modify the existing data in the ResourceSequence.txt file.

Related Topic

Customizing Siebel Migration Execution and Resource Sequencing

About the Process Flow for Migration Resources

The orchestration.json file defines the migration process flow for all the migration resources. The file has the following sections:

- **PreProcessing_Export:** This section contains the steps to be executed before the resources are exported.
- **PostProcessing_Export:** This section contains the steps to be executed after the resources are exported.

- **PreProcessing_Import:** This section contains the steps to be executed before the resources are imported.
- **PostProcessing_Import:** This section contains the steps to be executed after the resources are imported.
- **Export:** This section contains the resources and the corresponding steps to be exported.
- **Import:** This section contains the resources and corresponding steps to be imported.

Note: If you made any changes to the ResourceSequence.txt file, then those changes must be added to the orchestration.json file after you complete the upgrade process.

Each Migration resource has a section in the orchestration.json file that describes its process flow. The process flow is made up of a number of sub-processes or steps. Each step in the migration process flow has the following attributes:

- **ResType:** Specifies the type of resource for migration, such as, pre or postprocessing, Business Service Export, or Import. The value for ResType is Process. This attribute applies only for the PreProcessing_Export, PostProcessing_Export, PreProcessing_Import, and PostProcessing_Import resources.
- **PlanType:** Specifies the plan type for migration. Possible values are:
  - ["Export"]: This applies for Export Only Asynchronous Migration Execution.
  - ["Import"]: This applies for Import Only Asynchronous Migration Execution.
  - ["Export-Import"]: This applies only for Export and Import Synchronous Migration.
- **Business Service.** The name of the Siebel Business Service.
- **Step Name:** The name of the step in the Resource.
- **ApplyToResource:** Specifies the actual resource for migration. A valid value for ApplyToResource is one or more comma-separated Business Service names, as defined in the Import or Export sections. If this attribute has a value defined, then this step is executed only for the business services mentioned in the Import and Export sections. If this attribute is empty, then this step is executed for all business services. This attribute applies only for the PreProcessing_Export, PostProcessing_Export, PreProcessing_Import, and PostProcessing_Import resources.
- **Method.** The Siebel Business Service method.
- **Location.** The location where the Siebel Business Service will be executed. Values are either Source or Target.
- **LogMethod:** The method used to obtain the logs for the method.
- **InArg.** The Siebel Business Service input arguments.
- **OutArg.** The Siebel Business Service output arguments.
- **Async.** This section contains details if the method is Asynchronous.
  - **Async Business Service.** The Siebel Business Service Name.
  - **Async Method.** The Siebel Business Service method.
  - **Async InArg**. The Siebel Business Service input arguments.
  - **Async OutArg.** The Siebel Business Service output arguments.

An example of the step sequence in a migration process flow from the orchestration.json file follows:

```
"Steps": [
 {
 "ResType": "Process",
 "PlanType": ["Export"],
 "Business Service": "Application Migration Utility Service",
 "StepName": "GetWatermarkFromFile",
 "ApplyToResource": ["Migration Schema Service",
 "Migration Incremental Application Workspace Data Service",
 "Migration Incremental Runtime Repository Data Service"],
 "Method": "GetWatermarks",
 "Location": "Source",
 "LogMethod": "GetStatus",
 "InArg": ["sharedPath", "filename", "migrationid"],
 "OutArg": ["watermark"],
 "Async": {}
 ]
```

For more information about the (preprocessing and postprocessing) operations involved and methods invoked during the execution of a migration plan, see *Executing a Siebel Migration Plan*.

About the Siebel Migration Log Files

Environments: Development, production test, production.

Platforms: Windows, UNIX, IBM z/OS.

Siebel Migration creates log files that provide detailed information on the migration processes, including whether the migration succeeded or failed.

The Siebel Migration creates the following types of log files:

- **Migration log file.** The migration log file contains all the migration events, such as errors and warnings, for the Migration Application.

  The Siebel Migration Log file is located in the following directory:

  `<Application Interface Install Home>\applicationcontainer_external\logs\migration.log`

- **Business Service log file.** The Business Service log file is created in the EAI Object Manager log file when a Business Service is executed.

How to Locate Siebel Migration Resource Log Files

Siebel Migration resource files are stored in the following location:

Windows: `SIEBEL_FILESYSTEM/migration/<migration id>`

where:

- `SIEBEL_FILESYSTEM`: Indicates the location of the Siebel file system.
  - `migration:` Indicates the location of all the Siebel Migration files.
  - `inp:` Indicates the input file.
  - `rul:` Indicates the rule file.
  - `<migration id>:` The ID generated by the Siebel Migration Application when the user executes the migration plan.
    - `inp`: Contains a copy of the input file used for a migration execution.
    - `rul`: Contains a copy of the rule file used for a migration execution.

- `dat`: Contains all the data files generated by the export or moved from the source to target for import.
- `log`: Contains the schema file generated schema export or moved from source for import log file generated by the export or import.
- `schema`: Contains the schema file generated schema export or moved from source for import.
- `other`: Contains the web artifacts or file system artifacts for export and moved from the source for import.

Enabling SQL Tracing for Siebel Database Utilities

You can enable SQL tracing by setting the `DBUTIL_LOG_EVENTS` environment variable before you run any of the Siebel database utilities, for example, the ddlimp, dataimp, ddlsync, or repimexp utilities. These utilities are invoked by the Siebel Migration application, therefore the environment must be set up before running the migration plan. Depending on the operating system you are using, `DBUTIL_LOG_EVENTS` should be set as the System Environment variable (on Windows) or exported to the siebenv.sh shell script (on UNIX).

To enable SQL tracing for Siebel database utilities

1. On Windows operating systems, define a System Environment variable as shown in the following table.

Variable Name	Value
`DBUTIL_LOG_EVENTS`	`SQLParseAndExecute=5,SQLDBUtilityLog=3`

2. On UNIX operating systems, export the environment variable in the siebenv.sh shell script as follows:

```
export DBUTIL_LOG_EVENTS="SQLParseAndExecute=5,SQLDBUtilityLog=3"
```

You must restart the Siebel Server for the change to take effect.

About REST API Used for Migration Discovery and Execution

Environments: Development, production test, production.

Platforms: Windows, UNIX, IBM z/OS.

REST API requests are used for migration resource discovery and for migrating data from the source environment by exporting the data, transferring the data to the target environment, and importing the data into the target environment.

Siebel Migration includes the migration discovery service that is available to assist with discovering resources available for migration. The discovered services are listed in the Siebel Migration in the sequence in which they are executed during the migration process.

While creating a Siebel Migration plan, you can choose one or more services that are available. When you execute the Siebel Migration plan, those services are invoked to migrate the data.

The following table lists the migration resources that are available in the Siebel Migration.

Migration Service	Description	Supported Methods
Schema Service	Migrates the physical Siebel schema from the source environment to the target environment. When you use this service, the Siebel Migration prompts you to enter the Table Owner User Name, the Table Owner Password, and Database Encoding.	• Export • Import • GetWatermark • IsSchemaChanged • GetStatus
Runtime Repository Data Service	Migrates only the Runtime Repository from the source environment to the target environment. This includes Admin Data such as List of Values, Predefined Queries, etc. The migrated repository is named Migration Repository in the target environment. This includes Admin Data such as List of Values, Predefined Queries, etc. The user must select the name and version of the Workspace Branch. The default version will be the latest version. After the Siebel Migration is complete, you must change the Migrated Repository to Siebel Repository in the S_REPOSITORY table.	• GetRRInfo • GetWatermark • Export • Import • DBCheck • GetStatus
Incremental Runtime Repository Data Service	Identifies the version of the repository data that was previously migrated. This service takes all the changes from the previously migrated version and the latest version and migrates the data to the target environment. You must run the Schema Service along with the Incremental Runtime Repository Data Service. The Schema Service only runs if there is a change detected as part of the incremental migration. Otherwise, the Schema Service will not run even though it is part of the selection.	• Export • Import • GetWatermark • DBCheck • GetStatus
Incremental Application Workspace Data Service	Identifies the version that was previously migrated. This service takes all the changes from the previously migrated version to the latest version and migrates them to the target environment.	• GetWatermark • Export • Import • GetStatus
Application Data Service	This service migrates the data from the source environment to the target environment based on the tables listed in the `datamig.inp` file on the source environment. It is recommended that you review the tables configured in `datamig.inp` to ensure that it includes the correct tables.	• Export • Import • GetStatus

Migration Service	Description	Supported Methods
Application Data Service With Transformation	Migrates the data from the source environment to the target environment based on the tables listed in the **`datamig.inp`** file on the source environment. While exporting the data, this service uses the rule defined in the **`datamig.rul`** file and performs the transformation. The transformed data will be migrated to the target environment. It is recommended that you review the tables configured in **`datamig.inp`** and **`datamig.rul`** to ensure that they include the correct tables.	• Export • Import • GetStatus
File Prepare And Deploy Service	Identifies all the new or modified files and migrates the files to the target environment. On the target environment, the File Prepare And Deploy Service transfers the modified or new files to each Siebel Application Interface node defined in Siebel Management Console (SMC). This service keeps track of the files that are migrated for each target environment. The next time that the user runs this service, the service will check the modified files or newly created files from the previous migration. This service migrates the file artifacts from Siebel Application Interface and the file system. The File Prepare and Deploy Service reads the checksum values for all the web artifact files from the watermark file. The File Prepare and Deploy Service compares the checksum of the files present in the source web artifacts path. The File Prepare and Deploy Service takes files whose checksum does not match and generates an export package. The files that are not included in the watermark file are included in the export package. While importing the export package on the target environment, the existing files will be overwritten.	• Prepare • Deploy • GetStatus

Related Topic

REST API References for Migration Services

22 Data Preparation for Siebel Migration

Data Preparation for Siebel Migration

This chapter provides initial preparatory information for migrating your data with Siebel Migration. This chapter includes the following topics:

- *Process of Preparing Siebel Application Data for Migration*
- *Process of Transforming Data with Siebel Application Deployment Manager*
- *Customizing Siebel Migration Execution and Resource Sequencing*
- *Setting Up File Prepare and Deploy*

Process of Preparing Siebel Application Data for Migration

Environments: Development, production test, production.

Before you use Siebel Migration to migrate your data, you must perform the following tasks:

1. Configure and deploy your Migration Profile using Siebel Management Console. For more information about using Siebel Management Console to configure and deploy your Migration Profile, see *Siebel Installation Guide for Microsoft Windows* .
2. Create ADM Data Maps to transform the data with Application Deployment Manager. For more information about creating ADM Data Maps to transform the data with Application Deployment Manager, see *Process of Transforming Data with Siebel Application Deployment Manager* .
3. Customize Siebel Migration Execution and Resource Sequencing. For more information about customizing Siebel Migration Server Execution and Resource Sequencing, see *Customizing Siebel Migration Execution and Resource Sequencing*.
4. Setup File Prepare and Deploy. For more information, see *Setting Up File Prepare and Deploy*.

Process of Transforming Data with Siebel Application Deployment Manager

Application Deployment Manager (ADM) is a Siebel tool that you can use to deploy data from the source environment to the target environments. For more information about Application Deployment Manager, see *Siebel Application Deployment Manager Guide* .

For your data migration, you can use ADM to:

- Create ADM Projects that can be migrated by Siebel Migration. For more information about creating ADM projects, see Application Deployment Manager Guide.

When you create your ADM Project, you can configure the Sequence Number. The Sequence Number is populated in the Project Items. ADM migrates the Project Items in the sequence order according to the value of the Sequence Number parameter. If the sequence number is not provided, ADM will follow the default behavior.

- Create an ADM Data Map to transform data before importing into the target environment. For more information about creating ADM maps, see *Creating an ADM Data Map*.
- Associate your data map to your project item. For more information about associating data maps to ADM projects, see *Associating a Data Map to a Project Item*.

Creating an ADM Data Map

This topic provides procedures for creating an ADM Data Map.

This task is a step in *Process of Preparing Siebel Application Data for Migration*.

> **Note:** EAI Data Maps can be Workspace enabled in your development environment. If they have been Workspace enabled in your development environment and you are working in that environment, then you can only modify them in an editable Workspace. You do not need an editable Workspace to create and edit EAI Data Maps in your Production environment.

To create an ADM data map

1. If EAI Data Maps have been Workspace enabled, create or open an editable Developer Workspace.
2. Navigate to the Site Map, Administration - Integration, and then Data Map Editor.
3. In the Data Map Editor applet, create a new record.
4. Enter a Name for the data map.
5. Choose the Integration Object that is used in the Data Type for both the Source Object Name and the Target Object Name.
6. Click Auto-Map to populate the Integration Component Map and Integration Field Map fields.
7. In the Integration Component Map applet, select the Integration Component to transform the data.
8. In the Integration Field Map applet, select the Field to transform the data.
9. In the Source Expression column, enter the expression to transform the data. For more information about supported expressions, see Application Deployment Manager Guide.
10. Click Save.

Associating a Data Map to a Project Item

This topic provides procedures for associating an ADM Data Map to a Project Item.

This task is a step in *Process of Preparing Siebel Application Data for Migration*.

To associate a Data Map to a project item

1. Navigate to the Site Map, Application Deployment Manager, and then Deployment Projects.
2. Choose an existing Deployment Project or create a new Deployment Project and fill the necessary details. To create a new Deployment Project, see *Siebel Application Deployment Manager Guide* .
3. Choose an existing Project Item or create a new Project Item and fill the necessary details. To create a new Project Item, see Application Deployment Manager Guide.

4. Click the Data Map column on the Project Items applet and select the Data Map that will be used to transform the data.
5. Click Save.

Customizing Siebel Migration Execution and Resource Sequencing

Siebel Migration executes a set of business service methods for each migration resource. The execution sequence of these methods is defined by an external sequence text file, ResourceSequence.txt. The ResourceSequence.txt file is located in the following directory:

```
<Application Interface Home>\applicationcontainer_external\webapps\siebel\WEB-INF
```

Siebel Migration reads through this file during the execution process and executes the services in a sequential order. The ResourceSequence.txt file lists the names of the Business Services for each resource and the supported methods for Export, Import and Status.

You can customize Siebel Migration execution and resource sequencing by adding business services to the ResourceSequence.txt file. For more information about guidelines for adding business services to the ResourceSequence.txt file, see *About Migration Process Orchestration During the Siebel Migration Process*.

In the ResourceSequence.txt file, business services have the following structure:

```
<Service Name>,
<Method:Source/Target:Sync\Async:FT:trackingIdMethod:{Watermark Method Name|Source\Target|InputY\N}>,
<Method:Source/Target:Sync\Async:FT:trackingId:{Watermark MethodName|Source\Target|Input Y\N}>,
GetStatusMethod
```

Where:

- `<Service Name>` indicates the name of the business service.
- `<Method>` indicates the method name.
- `Source\Target` indicates whether the method will be executed on the migration source or target.
- `Sync\Async` indicates whether the method is sync or async.
- `<FT>` indicates whether a file transfer is required Y\N.
- `trackingIdMethod` indicates the MethodName whose trackingId should be the input for the GetStatus and Import methods.
- Watermark details:
  - Watermark Method Name
  - Source\Target
  - Input required Y\N
- GetStatusMethod indicates the status method for async methods.

Note: Existing business service entries and sequence must not be modified.

File Transfer Service is a another service which is not part of Migration discovery.

The details of the File Transfer Service are as follows:

```
[File Transfer Resource]
Dock Migration File Transfer Service,GetConnectString,ImportFile,GetStatus
```

Example of adding a new business service in the resource sequence file:

```
Migration Schema Service,Export:Source:Async:::Y,Import:Target:Async:Export::N,GetStatus
```

In this example:

- The Migration Schema Service is the Business Service Name with 2 async methods.
- The Export method should be run on Source and requires File Transfer.
- The Import method should be run on Target and does not require File Transfer.
- The GetStatus method is used to poll the status of these methods.

  For sync methods, GetStatusMethod is not required. Execution will continue based on the HTTP response code.

Setting Up File Prepare and Deploy

Siebel Migration File Prepare and Deploy service does the following:

- Migrates file artifacts from the source environment to the target environment.
- Migrates only file artifacts from Siebel Application Interface (AI) and the Siebel File System.
- If the target environment has multiple Application Interface nodes, it migrates all file artifacts to all the Application Interface nodes.
- Supports incremental file Migration (migrates only new or modified files).
- Uses the Checksum utility to check whether the file is modified or not.

The list of directories considered for Migration are configured in the filemig_config.properties file. The filemig_config.properties file is located in the following directory:

```
<Application Interface Home>\applicationcontainer_external\webapps
```

The filemig_config.properties file:

- Contains three properties:
  - inclusion-dir
  - exclude-dir
  - exclude-extension
- Considers all the files from the list of directories provided in the inclusion-dir property.
- Ignores all the directories mentioned in the exclude-dir from the include-dir only.
- Ignores the files whose extensions are part of exclude-extension property.

Sample filemig_config.properties file:

```
inclusion-dir=%ORACLE_HOME%/applicationcontainer_external/siebelwebroot/files, %FS_HOME%/att
exclude-dir=%ORACLE_HOME%/applicationcontainer_external/siebelwebroot/files/3rdParty exclude-extension=xps,
 pdf
```

Where:

- **`%ORACLE_HOME%`** is the AI installation directory.
- **`%FS_HOME%`** is the Siebel File System Path.

You can add multiple directories separated by a comma.

23 Data Migration Using Siebel Migration

Data Migration Using Siebel Migration

This chapter provides guidelines for performing a migration with Siebel Migration. This chapter includes the following topics:

- *Before You Begin Migrating with Siebel Migration*
- *Process of Using Siebel Migration to Migrate Data*
- *Asynchronous Migration Using Siebel Migration*
- *Migrating Repository Changes from Development to Production Environment (Siebel CRM 18.8 or Later)*
- *Migrating Configuration Data and Incremental Changes*
- *Managing Cross Version Migration*
- *Troubleshooting Data Migration Using Siebel Migration*

Note: Siebel Migration does not support migration of repository or Workspace data from RR environments to other RR environments. Migrating from Test to Production or UAT to Production is not permitted.

Before You Begin Migrating with Siebel Migration

Before you begin using Siebel Migration:

1. Review and follow the tasks listed in *Roadmap for Planning a Migration with Siebel Migration*
2. Start Siebel Migration with the following URL format:

   ```
   https://<hostname>:<port>/siebel/migration
   ```

3. Back up your target database before starting to migrate repository or data using Siebel Migration.
4. *Configure User Access to Siebel Migration Application*.

Configure User Access to Siebel Migration Application

The Siebel Migration Application depends on REST API calls to communicate with source and target environments. For security reasons, access to REST APIs is limited by Siebel responsibilities. In the out-of-the-box implementation, only the SADMIN user has access to these REST APIs. If another user needs to access the Migration Application, that user needs to be granted access via the procedure below. Failure to do so will result in the following error when trying to log in to the Siebel Migration Application: *Access is denied due to invalid credentials or insufficient privilege.*

To configure user access to Siebel Migration Application

1. In the source environment, navigate to Site Map, Administration - Application, Responsibilities, then Business Services.
2. Create a new responsibility named "Migration Users" and add the following business services to the new Migration Users responsibility.

Business Service	Business Service Methods
ADM Project Deploy Service	Get Status Migrate
Authentication Service For Migration	AuthenticateUser
ConnectionDataService	Delete Insert InsertOrUpdate QueryByExample Update
ExecutionDataService	Delete Insert InsertOrUpdate QueryByExample Update
HistoryDataService	Insert QueryByExample
Migration Application Data Service	Export GetStatus Import
Migration Application Data Service With Transformation	Export GetStatus Import
Migration Application Workspace Data Service	FullSeedExport FullSeedImport

Business Service	Business Service Methods
	GetFullSeedWatermark GetSeedCopyWatermark GetStatus InvalidateSeedCaches SeedCopyExport SeedCopyImport
Migration Design Repository Data Service	DBCheck Export GetStatus Import
Migration File Prepare And Deploy Service	Deploy GetStatus Prepare
Migration Incremental Application Workspace Data Service	Export GetStatus GetWatermark Import InvalidateSeedCaches
Migration Incremental Runtime Repository Data Service	DBCheck Export GetStatus GetWatermark Import

Business Service	Business Service Methods
Migration Runtime Repository Data Service	DBCheck Export GetRRInfo GetStatus GetWatermark Import
Migration Schema Service	DBCheck Export GetStatus GetWatermark Import IsSchemaChanged
PlanDataService	Delete Insert InsertOrUpdate QueryByExample Update

3. Clear the cache so that the changes will be propagated to the new responsibility.
4. Create a new user (for example, MIGUSER) and associate the Migration Users responsibility with the new (MIGUSER) user.
5. Log in to the Siebel Migration Application as MIGUSER and verify that login is successful.
6. When successful login is confirmed, repeat steps 1 to 5 in the target environment and test the login.

Process of Using Siebel Migration to Migrate Data

This procedure describes how use Siebel Migration for migration tasks. Siebel Migration tasks include:

1. Creating connections. For more information, see *Creating a Connection*.

2. Generating watermarks. For more information, see *Generating a Watermark*.
3. Creating migration plans. For more information, see *Creating a Migration Plan*.
4. Executing migration plans. For more information, see *Executing a Siebel Migration Plan*.
5. Executing full migration plans. For more information, see *Executing a Siebel Full Migration Plan*
6. (Optional) Aborting migration plans. For more information, see *Aborting a Running Migration Plan*
7. Reviewing migration log files. For more information, see *Viewing Migration Log Files*.
8. Reviewing migration history. For more information, see *Viewing Migration History*.

Note: The `dbcheck.exe` utility (that finds differences between the logical Siebel Schema and the physical database) has been removed from Migrations. Dbcheck is not a necessary part of a Migration. This allows Migrations to complete more quickly. Customer may run dbcheck whenever they desire to find any differences between the logical schema and the physical database.

Creating a Connection

This procedure describes how to use Siebel Migration to create a connection for a migration. Creating a connection registers a Siebel environment to the Siebel Migration. Each connection represents either a Source (development) or Target (production) environment to be used for migration. Numerous connections can be created for a migration.

This task is a step in *Process of Using Siebel Migration to Migrate Data*.

Note: The ability to enter database information (define database parameters) when creating a connection for Siebel Migration is available in Siebel CRM 20.3 Update and later releases. Prior to this, database information was captured when executing the migration plan.

To create a connection

1. In the Siebel Migration Application, click Connections in the navigation menu in the side panel to go to the Connections screen. You define the Source or Target of a migration plan in this screen.
2. Click New Connection, and in the New Connection window that appears:

   Note: The Name field and REST Endpoint field are mandatory fields; all other fields in the New Connection window are to be completed as required.

   a. Enter a name for the new connection in the Name field.
   b. Enter a valid endpoint for the connection in the REST Endpoint field.

      The format for a valid REST Endpoint is as follows:

      ```
      https://<hostname>:<port>/siebel/v1.0
      ```

      Where `siebel` can be replaced with any customized URL.
   c. If this connection will ever be used as the target of a migration plan, then in the Database Information Applicable for Target Environments only section of the New Connection window, enter the information described in the following table.

Field	Description
DB Schema Owner User ID	Enter the Database Table Owner user name.

Field	Description
Tablespace for Data	Enter the name of the tablespace where you want to import the tables. This field applies to Oracle and DB2 LUW. For any other database, this value will be ignored.
Tablespace for Index	Enter the name of the tablespace where you want to import the indexes. This field applies to Oracle and DB2 LUW. For any other database, this value will be ignored.
Tablespace for 16k Page	Enter the name of the 16k tablespace where you want to import the tables. This field applies to DB2 LUW only. For any other database, this value will be ignored.
Tablespace for 32k Page	Enter the name of the 32k tablespace where you want to import the tables. This field applies to DB2 LUW only. For any other database, this value will be ignored.
Database Encoding	Select one of the following options: - UNICODE Database - Non-UNICODE Database

3. To modify an existing connection:

   a. Click Edit (the pencil icon) next to the connection that you want to edit.
   b. In the Edit Connection window that appears, modify the fields as required—see the previous table for more information.

Generating a Watermark

A watermark is required from the target environment to see what artifacts were previously migrated from the source environment. Based on the information in the watermark, the various Migration Services can determine the delta artifacts that need to be packaged for incremental migration.

Siebel Migration obtains the watermark through a REST API call to the target environment. When the watermark is generated, it is placed under the "migration" folder in the Siebel File System, or Shared Package Location if defined, for the migration profile. When generating a watermark, the target must be a Runtime Repository environment. If the connection points to a development environment, then Siebel Migration throws an error.

This task is a step in *Process of Using Siebel Migration to Migrate Data*.

To generate a watermark

1. In the Siebel Migration Application, click Connections in the navigation menu in the side panel to go to the Connections screen.
2. Click Watermark (the waves icon).
3. Enter the watermark file name.

4. Click OK.

   After the watermark file has been created, a window opens with a message detailing the filename and path to the watermark file.
5. Copy the watermark file to the migration folder in the Siebel File System or Shared Package Location where the Siebel Migration Application, which will connect to the source environment, is running.

Scenario for Using a Watermark

Consider performing a full migration from a source DR to a test RR environment. The Integration Workspace in the source might be "int_May_2022" and on version 19. When a full migration is executed, the target RR environment will have a MAIN Workspace version of "0" and the information that it came from (int_May_2022) will also be stored.

At some point in the future, after a few features have been delivered and some bugs fixed, int_May_2022 might be on version 27 and you want to migrate *only* the changes to the test RR environment. To do this:

- You need to determine what was sent the last time that migration occurred. You do this by querying the target environment for the watermark.
- The RR environment responds (in this case) that it was last updated from the int_May_2022\19 Integration Workspace, and the migration execution on the DR environment packages up the correct contents accordingly.

Note: This is tracked in the target RR environment rather than in the DR environment because there is no way to know for certain if a particular export job was ever imported to the target environment. For example, you export some version X with the "intent" to import it into some target environment, but never actually do so. In the case of asynchronous migration, there is no way for the DR environment to know if the import ever occurred. Only the target environment knows for sure what was imported.

Types of Watermarks

There are several types of watermarks in a generated watermark file, including the following:

- Source Integration Workspace and Version for Workspace-enabled objects.
- Last Updated Date for objects that are not Workspace-enabled (such as schema changes).
- Checksums for files when using the File Prepare and Deploy service.
- The target environment's binary version – this allows for proper migration when the target environment is on a lower Siebel CRM monthly update version.

Creating a Migration Plan

This procedure describes how to use Siebel Migration to create a migration plan for a migration. The procedure involves selecting connections, including Source and Target environments, and resources that you want to migrate as part of the migration plan. In Siebel Migration, you can select Database Utilities, ADM Projects or File Prepare and Deploy services. When you run a migration plan, Siebel Migration exports the migration data from the Source environment and imports it into the Target environment.

This task is a step in *Process of Using Siebel Migration to Migrate Data*.

To create a migration plan

1. In the Siebel Migration Application, click Migration Plans in the navigation menu in the side panel to go to the Migrations Plans screen.

2. Click New Migration.
3. Enter the Name of the Migration Plan and a Description.
4. Select and move the Source Connection & Target Connection onto the Flow Chart.

   An intersection of Migration Resources (between Source & Target) for Database Utilities and only from Source for ADM Projects will be displayed.
5. Select the Migration Resource that will be executed as part of the migration plan.

Siebel Migration Plan Dependencies

The Siebel Migration Discovery services that appear in Siebel Migration depends on selections that you make. The following table outlines the Siebel Migration plan dependencies.

If You Select this Resource...	Read Only Resources	Auto Selected Resources
Application Workspace Data Service	• Incremental Repository Service • Incremental Application Workspace Data Service	• Schema Service • Runtime Repository Service
Increment Repository Data Service	• Runtime Repository Service • Application Workspace Data Service	• Schema Service
Runtime Repository Data Service	• Incremental Repository Service • Incremental Application Workspace Data Service	• Schema Service • Application Workspace Data Service
Incremental Application Workspace Data Service	• Runtime Repository Data Service • Application Workspace Data Service	• None

Executing a Siebel Migration Plan

This procedure describes how to use Siebel Migration to execute a migration plan for a database migration.

This task is a step in *Process of Using Siebel Migration to Migrate Data*.

To execute a migration plan

1. In the Siebel Migration Application, click Execution in the navigation menu in the side panel to go to the Execution screen.

   The following table shows the information that is available for each migration plan in the Execution screen.

Field	Description	Example Value
Name	Migration plan name	Data Export

Field	Description	Example Value
Description	Migration plan description	Data Export Migration
Status	Migration plan status, which can be one of the following: Null or Running.	Running
Action	Migration plan action, which can be one of the following: Run or Stop.	Run
Source	Migration plan source integration branch.	DEV
Target	Migration plan target integration branch.	TEST
Archive ID	Migration plan archive ID. This is read-only and will be populated only for migration plans that are currently executing.	88-1WHQVG
Start Date	Start timestamp for migration operation.	2022-03-03 20:59:13
End Date	End timestamp for migration operation.	2022-03-03 21:10:10
Package Filename	Migration plan package filename.	TEST_2022_03_03

2. Select the migration plan that you want to execute, and then click Run in the Action field.
3. Click OK when prompted with the following (or similar) message:

```
Are you sure you want to execute '<Name_of_Plan>' migration plan?
```

4. In the Execution Details window that appears, complete the fields as required and then click OK.

   Depending on the configuration of the migration plan that you are executing (that is, the action and resources or services selected in the migration plan), the Execution Details window will prompt you to enter different information, as follows:

   a. **Package Filename.** Enter the name of the package file (.zip) to export or import from. This field applies only to Export Only and Import Only migration plans.
   b. **Watermark.** Enter the name of the watermark file (.txt). This field applies to Export Only migration plans that contain Incremental Runtime Repository Data Service (IRR) or File Prepare & Deploy.
   c. **Workspace Branch Name and Workspace Version**. These fields apply only during an Incremental Runtime Repository migration. For Export and Import (Sync) migration plans, these fields are automatically populated based on the Waternark file.

      Note: In the case of Export Only migration plans, you must click Get Workspace Details to populate these fields. The value in the Workspace Version Number field defaults to the very latest version, but you can select a different version if required. The Workspace Version Number is the range between the next version of the Target and the latest available version on the Source.

   d. **Database Information for Target Environment.** The fields in this section apply to the following migration plans that contain a Schema Service: Import Only, Export and Import (Sync). Complete the fields in this section as follows:

      - If you have already defined the database parameters for your target connection, then enter the password of the target connection's Database Table Owner in the DB Schema Owner Password field — all other database information in this section will be read-only. If you have already encrypted the DB Schema Owner Password using the *encryptstring.exe* utility and have entered that encrypted password into the DB Schema Owner Password field, select the *Pre-Encrypted* check box (Set it to true). This tells the Migration Application that the password it will be using has already been encrypted.

        Note: You can still enter a plain-text password for the DB Schema Owner Password as you have always done. In this case, don't select the Pre-Encrypted check box, instead enter your plain-text password in the DB Schema Owner Password field.

      - If you have not already defined the database parameters for your target connection, then enter the information described in the following table.

        Note: If you have to specify the database parameters during execution of a migration plan, then the database information will be saved for the connection as well, and you will not have to specify the database parameters for any subsequent executions of the migration plan.

Field	Description
DB Schema Owner User ID	(Required) Enter the database schema owner user ID for the target connection.
DB Schema Owner Password	(Required) Enter the database schema owner password for the target connection.

Field	Description
Tablespace for Data	Enter the name of the tablespace on the target where you want to import the tables. This field applies to Oracle and DB2 LUW. For any other database, this value will be ignored.
Tablespace for Index	Enter the name of the tablespace on the target where you want to import the indexes. This field applies to Oracle and DB2 LUW. For any other database, this value will be ignored.
Tablespace for 16k Page	Enter the name of the 16k tablespace on the target where you want to import the tables. This field applies to DB2 LUW only. For any other database, this value will be ignored.
Tablespace for 32k Page	Enter the name of the 32k tablespace on the target where you want to import the tables. This field applies to DB2 LUW only. For any other database, this value will be ignored.
Database Encoding	(Required) Select one of the following options: ◦ UNICODE Database ◦ Non-UNICODE Database
Pre-Encrypted	TRUE/FALSE, based on your way of passing schema password. ◦ **TRUE** for encrypted value of Schema owner Password(check this flag). ◦ **FALSE** for Plain text of schema owner password.

5. In the Name field, expand the '<Name_of_Plan>' migration plan to show all resource task details for the migration plan.

   The following information is available for each resource task in the migration plan: Resource Name, Operation, Seq Num, Status, Action, Log, Start Date, and End Date. For more information about these fields, see *Viewing Migration Log Files*.
6. Click Refresh in the Action field of any of the resource records while the migration plan is running to refresh the migration plan and get the latest status for all resource tasks.
7. Click Log in the Log field while the migration plan is running to view the log details for a resource task

Renaming Repositories After Full Migration

Note: No other mechanism for renaming repositories in a Runtime Repository environment is supported. Examples of unsupported methods include using Siebel Tools, Web Tools, or direct SQL.

After a full migration execution using the Runtime Repository Data Service, your target environment will have (at least) two repositories – "Migrated Repository" and "Siebel Repository". To switch to using the newly migrated repository, you must stop all Siebel services and rename them using the siebdevcli utility:

Note: Do not perform this procedure after an Incremental Runtime Repository migration.

To rename repositories after a full migration

1. Stop all Siebel services.
2. Rename Siebel services using the siebdevcli utility as follows:

   ```
   siebdevcli $SIEBEL_HOME\siebsrvr\bin\enu\<siebel.cfg> /l <language_code> /u <username>
   /p ******** /d <DataSourceName> /SwitchRepository /SRCurrent <current repository> /SRNew <new re
   pository> /SROld <old repository name>
   ```

   Where:

   - `<siebel.cfg>` is the name of the configuration file – typically "siebel.cfg" located in the `$SIEBEL_HOME\siebsrvr\bin\enu` directory. This file must contain correct database connection parameters. For more information about the siebel.cfg file, see *Modifying siebel.cfg Before Upgrading Siebel Database* and *Troubleshooting Database Configuration*.
   - `<language_code>` is the language in which siebdevcli should run (such as "ENU"). This will determine the language used for log files and errors.
   - `<username>` is any Siebel Administrator username (for example, SADMIN).
   - `<password>` is the password for the username specified with `/u`.
   - `<DataSourceName>` is the data source entry from the siebel.cfg file (typically "ServerDataSrc").

     Note: The ServerDbODBCDataSource parameter under the [Siebel] section in the siebel.cfg file must point to the correct data source – that is, the ODBC_DSN on the application server. For example: `ServerDbODBCDataSource = "siebel_DSN"`.

   - `<current repository>` is the name of the repository you are trying to replace (typically "Siebel Repository").
   - `<new repository>` is the name of the recently migrated repository (typically "Migrated Repository").
   - `<old_repository_name>` is the name you want to give to the existing <current repository> (for example: "Old Repository").

   For example:

   ```
   siebdevcli /c $SIEBEL_HOME\siebssrvr\bin\enu\siebel.cfg /l ENU /u SADMIN /p ******
   /d ServerDataSrc /SwitchRepository /SRCurrent "Siebel Repository"
   /SRNew "Migrated Repository" /SROld "Old Repository"
   ```

   After executing this command, your old "Siebel Repository" will be renamed "Old Repository" and your recently migrated repository will be named "Siebel Repository".

   Upon restarting the server, the Application Object Manager will use the updated repository definition.

Preprocessing and PostProcessing for Migration Execution

The operations involved during the preprocessing and postprocessing of any migration plan execution include for example, Package, Unpackage, Transfer the File (for Sync migration), Create Manifest, GetConnectString, and so on. A number of preprocessing and postprocessing methods are invoked during the execution of a migration plan. The following table describes these methods, which are in the orchestration.json file.

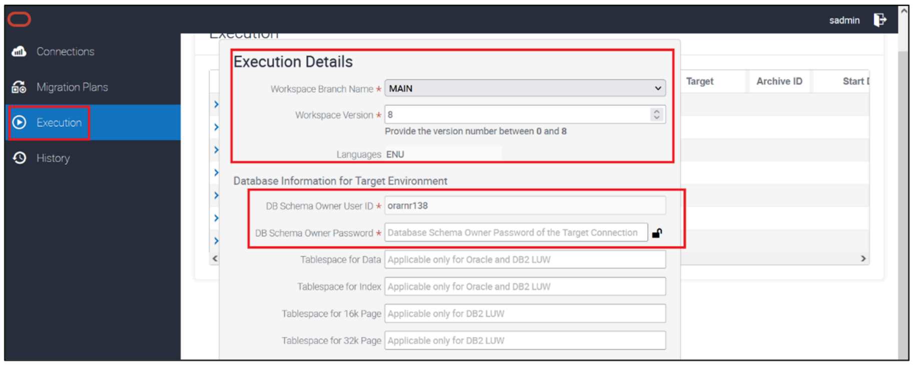
Resource	Operation	Purpose	Applies to Resource	Mode
PreProcessing_Export	GetWatermarks	Gets the watermark details from the watermark file specified during migration execution.	Migration Schema Service Migration Incremental Application Workspace Data Service Migration Incremental Runtime Repository Data Service	Asynchronous
PreProcessing_Export	GetWaterMark	Gets the watermark details by invoking the REST request to the Target.	Migration Schema Service Migration Incremental Application Workspace Data Service Migration Incremental Runtime Repository Data Service	Synchronous
PostProcessing_Export	CreateManifest	Creates the manifest file, which includes all the details of the resource being exported. The manifest file is used to validate credentials during the asynchronous migration import.	All services	Synchronous Asynchronous
PostProcessing_Export	preparePackage	Creates a compressed package file that contains all the exported resources.	All services	Synchronous Asynchronous
PostProcessing_Export	GetConnectString	Gets the SynchMgr Object Manager Connect String from Source. This will be used by ImportFile to transfer the file from Source to Target. This operation applies only if the Migration Package Location in the migration profile in Siebel Management Console is not specified.	All services except ADM	Synchronous
PostProcessing_Export	ImportFile	Transfers the package from Source to Target. This operation applies only if the Migration Package Location in the migration profile in Siebel Mobile Console is not specified.	All services except ADM	Synchronous
PreProcessing_Import	unPackage	Decompresses the exported package.	All services	Synchronous

Executing a Siebel Full Migration Plan

This topic describes how to use Siebel migration to execute a full migration plan.

As migration plans are intended to be re-usable, they define only the source, target, and the resources (such as Schema Service, ADM Projects, and so on). The selection of **Source Workspace** and **Version** are selected at the time of execution.

When you execute a synchronous full migration plan, the migration application prompts for the source workspace, along with the target database schema owner and password.



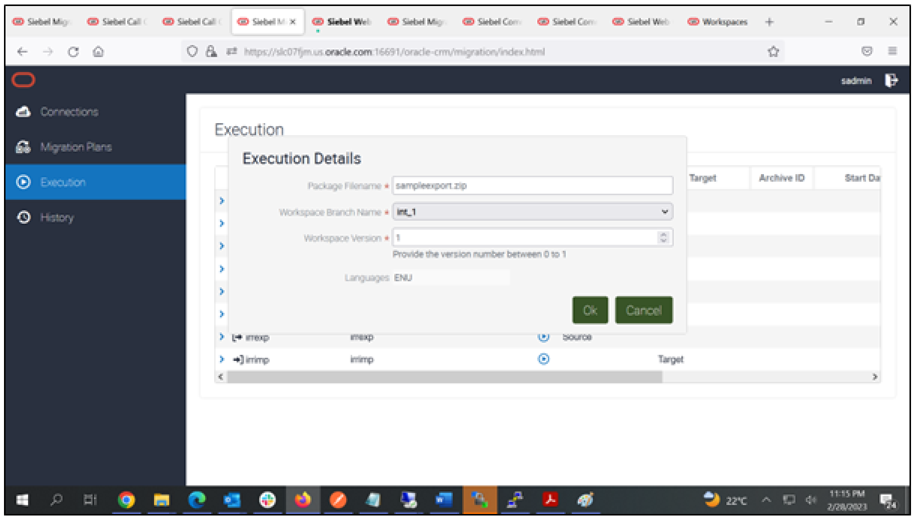
Execution Details

Workspace Branch Name *	MAIN
Workspace Version *	8
	Provide the version number between 0 and 8
Languages	ENU

Database Information for Target Environment

DB Schema Owner User ID *	orarnr138
DB Schema Owner Password *	Database Schema Owner Password of the Target Connection
Tablespace for Data	Applicable only for Oracle and DB2 LUW
Tablespace for Index	Applicable only for Oracle and DB2 LUW
Tablespace for 16k Page	Applicable only for DB2 LUW
Tablespace for 32k Page	Applicable only for DB2 LUW

For an asynchronous full migration export job, the prompt is similar, but doesn't require the target schema owner and password, as shown in this image.



Execution

Execution Details

Package Filename *	sampleexport.zip
Workspace Branch Name *	int_1
Workspace Version *	1
	Provide the version number between 0 to 1
Languages	ENU

irrexp	irrexp	Source	
irrimp	irrimp		Target

Creating and Executing Migration Plans Through REST

This change affects REST calls to both create and execute migration plans for full migration. Information about the workspace name and version are passed to the execution call.

> **Note:** You don't have to change any existing scripts that create full migration plans that include the parameters *integrationBranchName* (Workspace Branch Name) and *irrEndVersion* (Workspace Version). This information will now be ignored when you create a migration plan for full migration. You simply must adjust the execution plan to include this information.

The new JSON payload for executing a full migration plan includes the two parameters required to specify the Workspace and Version.

Example of synchronous migration plan execution:

```
{
 "integrationBranchName": "MAIN",
 "schemaUser":"orarnr138",
 "schemaPassword":"TVNTUUw=",
 "watermarkFilename":"frr_watermark.txt",
 "irrEndVersion":"7",
 "packageFilename":"frr_export_package.zip",
 "isUnicodeDatabase":"Y"
}

Example of asynchronous Migration Plan Execution (export):
{
 "integrationBranchName": "MAIN",
 "irrEndVersion":"7",
 "packageFilename":"frr_export_package.zip",
}
```

Aborting a Running Migration Plan

This procedure describes how to stop a running migration plan. This is useful if, for example, you selected the wrong source integration branch when executing the migration plan. When you stop a running migration plan, the following happens:

- All resource tasks within the migration plan will also end.
- The status on any currently running resources changes to Stopped and the migration plan's status changes to Stopped, and this information is moved to the History screen. For more information, see *Viewing Migration History*.
- The resource that is running in the Source or Target environment will continue to run until it completes – there is no way to stop this nor to rollback the resource (that is currently running or already completed).

To abort a running migration plan

1. In the Siebel Migration Application, click Execution in the navigation menu in the side panel to go to the Execution screen.
2. Identify and select a migration plan (for example, Data Export) with a status of Running that you want to stop.
3. Click Stop in the Action field.

4. Click OK when prompted with the following (or similar) message:

```
Are you sure you want to stop the execution?
```

5. When the stop operation completes, click OK when prompted with the following (or similar) message:

```
The execution for the Migration Plan '<Name_of_Plan> - YYYY-MM-DD Hr:Min:Sec'
was successfully stopped.
```

   In the Execution screen, notice that the '<Name_of_Plan>' migration plan will stop running and the Run (or play) icon in the Action field will be enabled for the record so that you can execute the plan again.
6. Click History in the navigation menu in the side panel to go to the History screen.

   Notice that the '<Name_of_Plan>' migration plan that you stopped in step 3 has a status of Stopped. For more information, see *Viewing Migration History*.
7. (Optional) To re-execute the migration plan that you stopped in step 3, return to the Execution screen, select the migration plan, and then click Run in the Action field.

Viewing Migration Log Files

This procedure describes how to use Siebel Migration to view the log files for a migration plan. You can access and view log files from the following screens:

- Execution screen – for more information, see the following procedure.
- History screen (after a migration completes) – for more information, see *Viewing Migration History*.

This task is a step in *Process of Using Siebel Migration to Migrate Data*.

To view migration log files

1. In Siebel Migration, click Execution in the navigation menu in the side panel to go to the Execution screen.

   The following information is available for migration plans in the Execution screen: Name, Description, Status, Action, Source, Target, Archive ID, Start Date, End Date, and Package Filename. For more information about these fields, see *Executing a Siebel Migration Plan*.
2. In the Name field, expand the '<Name_of_Plan>' migration plan to show all resource task details for the migration plan.

   The following table shows the information that is available for each resource task in the migration plan.

Field	**Description**	**Example Value**
Resource Name	Resource task name – examples include the following for a Data Export migration plan: Application Data Service, PostProcessing_Export, and so on.	Application Data Service
Operation	Resource task operation – examples include the following: ◦ Export (for Application Data Service resource)	Export

Field	Description	Example Value
	◦ CreateManifest (for PostProcessing_Export resource) ◦ preparePackage (for PostProcessing_Export resource).	
Seq Num	Sequence number (1, 2, 3, and so on) indicating the order in which the resource tasks will be processed.	1
Status	Resource task status, which can be one of the following: Not Started, Running.	Not Started
Action	Refresh	Refresh
Log	Log	Log
Start Date	Start timestamp for resource task operation.	2022-03-03 20:59:13
End Date	End timestamp for resource task operation.	2022-03-03 21:10:10

3. Click Log in the Log field to view the log details for a resource task.

   You can review logs for the following resource operations:

   - Runtime Repository Data Service: GetWatermark operation.
   - Application Workspace Data Service: GetFullSeedWatermark and GetSeedCopyWatermark operations.
   - Incremental Application Workspace Data Service: InvalidateSeedCaches operation.
   - File Prepare and Deploy: generateWatermark, readwatermarkfile, writewatermarkfile operations.

Reviewing Execution Log Messages

Log messages typically record the Archive Id, Resource name, and the Operation name in the log file along with the log message. After an Archive Id is generated, all corresponding log messages will contain the Archive Id.

Log files are generated by the Siebel Migration Application but they are not visible in the migration UI.
If errors are encountered during migration execution, then review the migration.log file located in the `applicationcontainer_external\logs` folder on the server where migration is hosted to determine why the migration failed and to decide how to resolve the issue.

- To find and review the logs generated for a particular migration execution, search for the appropriate Archive Id in the migration.log file.
- To find and review the logs generated by a specific Resource under an Archive Id, search for the Resource name in the migration.log file.
- To find and review the logs generated by a specific Operation under a Resource Name, then search for the Operation in the migration.log file.

> **Note:** Migration logs are stored in the migration server's application container's (Apache Tomcat) log location: `<AI_ROOT>\applicationcontainer_external\logs`.

Once Siebel Migration starts executing operations that are part of a resource in the migration plan, all resulting log messages will contain the appropriate Archive Id, Resource name, and Operation name as shown in the following examples.

Example Log 1: Archive Id is in Brackets

```
[DEBUG] 2018-10-25 04:41:24.203 [Thread-33] Migration - com.siebel.migration.server.Transaction:callExecutor
 [88-1VBW0R] Exporting resource {"name":"Migration Schema Service","URI":["https:\/\/
slc04ovj.us.oracle.com:9001\/siebel\/v1.0"],"Source":"https:\/\/slc04ovj.us.oracle.com:9001\/
siebel\/v1.0","Target":"https:\/\/slc07fnj.us.oracle.com:9001\/siebel\/v1.0","Steps":
[{"StepName":"Export","Business Service":"Migration Schema Service","Method":"Export","InArg":
["migrationid"],"OutArg":["trackingid"],"Location":"Source","Async":{"Async Business Service":"Migration
 Schema Service","Async Method":"GetStatus","Async InArg":["migrationid","trackingid"],"Async
 OutArg":["error","getlog","log","status"]}}],"TransId":"88-1VBW0R","rowId":
["88-1VBW0S"],"Watermark":"","ActivationDate":""}
[DEBUG] 2018-10-25 04:41:24.212 [Thread-33] Migration -
 com.siebel.migration.server.ProcessFlow:<init>[88-1VBW0R] Entered process flow for resource : Migration
 Schema Service
```

Example Log 2: Archive Id and Resource Name are in Brackets

```
[DEBUG] 2018-10-25 04:41:24.214 [pool-4-thread-2] Migration -
 com.siebel.migration.server.ProcessFlow:getResources [88-1VBW0R] [Migration Runtime Repository Data
 Service] Entered get resources for resource : Migration Runtime Repository Data Service
[DEBUG] 2018-10-25 04:41:24.216 [pool-4-thread-1] Migration -
 com.siebel.migration.server.ProcessFlow:getResources [88-1VBW0R] [Migration Schema Service] Entered get
 resources for resource : Migration Schema Service
[DEBUG] 2018-10-25 04:41:24.216 [pool-4-thread-3] Migration -
 com.siebel.migration.server.ProcessFlow:getResources [88-1VBW0R] [Migration Application Workspace Data
 Service] Entered get resources for resource : Migration Application Workspace Data Service
```

Example Log 3: Archive Id, Resource Name and Operation are in Brackets

```
[DEBUG] 2018-10-25 04:41:24.234 [pool-4-thread-2] Migration -
 com.siebel.migration.server.Transaction:invokeRestCall [88-1VBW0R] [Migration Runtime Repository Data
 Service] [GetWatermark] Input PostData : {"body":{"workspace":"MAIN","language":"ENU","version":"0"}}
[DEBUG] 2018-10-25 04:41:24.251 [pool-4-thread-3] Migration -
 com.siebel.migration.server.ProcessFlow:getResources [88-1VBW0R] [Migration Application Workspace
 Data Service] [GetFullSeedWatermark] Argument Map constructed : {migrationid=88-1VBW0R, password=,
 workspace=MAIN, fileName=Migration_88-1VBW0R.txt, filename=, watermark=, isUnicodeDatabase=Y, sharedPath=,
 language=ENU, ActivationDate=, version=0, username=}
```

Viewing Migration History

This procedure describes how to use Siebel Migration to view the history details for a migration plan execution. The History screen keeps an ongoing log of the success or failure of each attempted migration plan execution, including resource task operations, along with the details of what happened. In addition, note the following:

- When a resource task fails in a migration plan, the Migration Application will not execute any subsequent resource tasks. The status on the migration plan and the failed resource task changes to "error" (the status on any unexecuted resource tasks changes to "Not Started"), and this information is moved to the History screen.
- If a migration plan execution fails and is left in a 'Running' state, then the execution instance is automatically marked as a failure and moved to the History screen. In the Execution screen, note that the migration plan will

stop running and the Run (or play) icon in the Action field will be enabled for the record so that you can start the plan again.

> **Note:** The cleanup described here only occurs upon restart of the applicationcontainer hosting that particular Migration Application.

- All log data in the History screen is persistent until it is specifically removed by administrators – for more information, see *Query Migration History* and *Cleanup Migration History*.

This task is a step in *Process of Using Siebel Migration to Migrate Data*.

To view migration history

1. In Siebel Migration, click History in the navigation menu in the side panel to go to the History screen.

   The following table describes the information that is available for each migration plan execution in the History screen.

Field	**Description**	**Example Value**
Name	Migration plan name	Data Export
Description	Migration plan description	Data Export Migration
Status	Migration plan status, which can be one of the following: Success, Aborted, Error, or Stopped.	Aborted
Source	Migration plan source integration branch.	DEV
Target	Migration plan target integration branch.	TEST
Archive ID	Migration plan archive ID, which is unique to a given execution of a migration plan.	88-1WHQVG
Start Date	Start timestamp for migration operation.	2022-03-03 20:59:13
End Date	End timestamp for migration operation.	2022-03-03 21:01:10
Package Filename	Migration plan package filename.	TEST_2022_03_03

2. In the Name field, expand the '<Name_of_Plan>' migration plan to show all resource task details for the migration plan.

   The following information is available for each resource task in the migration plan: Resource Name, Operation, Seq Num, Status, Action, Log, Start Date, and End Date. For more information about these fields, see *Viewing Migration Log Files*.
3. Click Log in the Log field to view the log details for a resource task.

Query Migration History

The following procedure shows how to query migration history data and filter the data shown in the History screen.

To query migration history

1. In Siebel Migration, click History in the navigation menu in the side panel to go to the History screen.
2. Click Query (the magnifying glass icon) to filter the records in the History view.

   The History Query form opens, where you can filter on the following fields: Name, Description, Status, Source, Target, Archive ID, Start Date, End Date, Package Filename. For more information about these fields, see *Viewing Migration History*.
3. For example, to search for all migration plan executions that were aborted:
   a. On the History Query form, enter the criteria shown in the following table.

Field	**Operation**	**Value**
Status	Equal (==)	Abort

   b. Click Go.

      All records that match this criteria are returned in the History screen.
4. For example, to delete all successful migration plan execution records from January 2022:
   a. On the History Query form, enter the criteria shown in the following table.

Field	**Operation**	**Value**
Status	== (Equal)	Successful
Start Date	>= (Greater than or equal to)	2022-01-01 00:00:01
End Date	<= (Less than or equal to)	2022-01-31 23:59:59

b. Click Go.
c. Select the check box beside the Name field of all returned records.
   - To select all visible records on the current page, click the check box beside the Name column heading.
   - Use the following navigation buttons to move to the next page or previous page: First Record Set, Previous Set, Next Record Set, Last Record Set.
d. Click Delete and then click OK when prompted to delete the records.

Cleanup Migration History

All log data in the History screen is persistent until it is specifically removed by administrations. To prevent information in the History screen from getting unwieldy, administrators can cleanup migration history by removing history records as shown in the following procedure.

To cleanup migration history

1. In Siebel Migration, click History in the navigation menu in the side panel to go to the History screen.
2. Click Query (the magnifying glass icon) to filter the records in the History view – for more information, see *Query Migration History*. This step is optional.
3. Select the check box beside the Name of each history record that you want to remove from the History screen.
   - To select all visible records on the current page, click the check box beside the Name column heading.
   - Use the following navigation buttons to move to the next page or previous page: First Record Set, Previous Set, Next Record Set, Last Record Set.
4. Click Delete and then click OK when prompted with the following (or similar) message:

   ```
   You have selected the below records. Do you want to delete them?
   '<Name_of_Plan> - YYYY-MM-DD Hr:Min:Sec'
    ...
   ```

   Clicking OK does the following:
   - Removes the selected record from the History screen.
   - Deletes all database records related to the migration plan execution in the selected record.
   - Deletes all associated log files and other artifacts (for example, data files in the File system belonging to the respective Source and Target connections) related to the migration plan execution in the selected record.

     Note: The deletion of files in the file system is an attempt. There are a number of reasons why deletion may fail. Consider, for example, a situation where you run synchronous migration from a Migration Application hosted in the Source environment and then later want to clean it up, but only the Source is available; to clean up the records, you attempt to send the delete request to the Target environment, but it fails because the Siebel Server is not up and running.

Asynchronous Migration Using Siebel Migration

You can use Siebel Migration to plan and carry out an asynchronous migration. An asynchronous migration essentially involves creating an asynchronous migration plan for the source environment and another (separate) asynchronous migration plan for the target environment. Migration activities that must be completed on the source environment can then be carried out independently of the migration activities that must be completed on the target environment. For asynchronous migration to work, both source and target environments must be in a Repository Upgraded state, otherwise migration execution will be performed in synchronous mode.

According to the source and target environment state, the following table outlines the supported migration scenarios. These migration scenarios are supported, using either the Siebel Migration Application or REST API, in Siebel CRM 18.11 Update and later releases.

Source and Target State	Export Only Migration	Import Only Migration	Export and Import Migration
Source is Repository Upgraded, Target is not.	Supported	Not Supported	Supported
Both Source and Target are Repository Upgraded.	Supported	Supported	Supported
Source is not Repository Upgraded, Target is Upgraded	Not Supported	Not Applicable	Supported

Note: A Repository Upgraded state means that the environment (repository and schema) has been upgraded to Siebel 18.8 Update or later release of Siebel CRM.

CAUTION: It is recommended that you back up your target database before starting to migrate repository or data using Siebel Migration.

The steps to perform an asynchronous migration using Siebel Migration are outlined in the following procedure and involve the following:

- Generating a watermark if required.
- Creating and executing an export only migration plan to export resources from the source environment.
- Creating and executing an import only migration plan to import (exported) resources to the target environment.

To perform an asynchronous migration using Siebel Migration

1. Generate a watermark if required. To export an Incremental Runtime Repository, you must get the Watermark file from the target environment where you are planning to import the exported data.

   > **Note:** You must ensure that the watermark you use is the latest watermark. If you export the repository using an old watermark, then the import will fail with an error message similar to the following: Watermark does not match the exported resources watermark.

   a. Navigate to the Connections tab in Siebel Migration.
   b. Click Watermark (the waves icon) next to the target connection (that is, the target environment where you want to import the exported data).
   c. In the dialog that appears, enter the Watermark file name and click OK.

      If the Migration Package Location is configured in the Migration Profile in Siebel Management Console, then the Watermark file is generated in the Migration Package Location path. Otherwise the Watermark file is generated in the `<File System>\migration` folder on the selected connection.
2. Create an export only migration plan on the source environment.

   a. Navigate to the Migration Plan tab in Siebel Migration.
   b. Click New Migration and then enter the Name of the migration plan and a Description.
   c. Select and move the connection, where you want to export data from, to the Source field.
   d. Select the resources that you want to export.
   e. Save the source migration plan.
3. Create an import only migration plan on the target environment.

   a. Navigate to the Migration Plan tab in Siebel Migration.
   b. Click New Migration and then enter the Name of the migration plan and a Description.
   c. Select and move the connection, where you want to import data to, to the Destination field.
   d. Select the resources you want to import.
   e. Save the target migration plan.
4. Execute the export only migration plan on the source environment.

   a. Navigate to the Execute tab in Siebel Migration.
   b. Go to and select the migration plan that you created in Step 2 and want to export, and then click Run (the play icon) in the Action column.
   c. Complete the fields in the Execution Details Window that appears as required, and then click OK. For information on how to complete the fields in the Execution Details Window, see *Executing a Siebel Migration Plan*.

      Siebel Migration starts to export all the resource data simultaneously. After all data has been exported, it is then packaged into a package file. If the Migration Package Location is configured in the Migration Profile in Siebel Management Console, then the package file will be created in the Migration Package Location path. Otherwise, the package file will be created in the `<File System>\migration` folder on the source machine.

      The package file contains a list of all the resources that were exported. It also contains the Watermark information if you have chosen the Incremental Runtime Repository or Incremental Workspace Data resource in the migration plan.

5. Execute the import only migration plan on the target environment:

   a. Navigate to the Execute tab in Siebel Migration.
   b. Go to and select the migration plan that you created in Step 3 and want to import, and then click Run (the play icon) in the Action column.
   c. Complete the fields in the Execution Details Window that appears as required, and then click OK. For information about how to complete the fields in the Execution Details Window, see *Executing a Siebel Migration Plan*.

   Before executing the import only migration plan, you must ensure and do the following:

   - Ensure that the package file (created in Step 4) is accessible to the target connection where you want to import the exported data. If the Migration Package Location is configured in the Migration Profile in Siebel Management Console, then the package file must be in the Migration Package Location path.
   - If the Migration Package Location is not configured, then you must copy the package file (created in Step 4) from the source to the target environment's `<File System>\migration` folder.

   Note: Before the import starts, Siebel Migration ensures that the resources selected in the import only migration plan match the resources selected in the export only migration plan. If the resources do not match, then an error message appears and the import will not start. Similarly, if the Incremental Runtime Repository or Incremental Workspace Data is part of the migration plan, then the watermark used during the export must match the watermark in the target environment. Otherwise , an error message appears and the import will not start.

   Note: The File System path must be accessible by all Siebel Servers and application interface machines.

Migrating Repository Changes from Development to Production Environment (Siebel CRM 18.8 or Later)

Once the development (source Design Time Repository/DR) environment is upgraded to the Siebel 18.8 Update or later release binary and repository, you must perform the steps in the following procedure to migrate the changes to the production (target Runtime Repository/RR) environment. You can use the asynchronous migration feature to do this only if both source (development) and target (production) environments are in a Repository Upgraded state. If either (source or target) environment is not in a Repository Upgraded state, then the migration execution will be performed in synchronous mode.

Note: A Repository Upgraded state means that the environment (repository and schema) has been upgraded to Siebel 18.8 Update or later release of Siebel CRM.

CAUTION: It is recommended that you back up your target database before starting to migrate repository or data using Siebel Migration.

To migrate Siebel 18.8 Update or later release repository changes from a development (source) to a production (target) environment

1. Create a connection to the target environment.
   For more information about creating a connection, see *Creating a Connection*.
2. Create a migration plan with Incremental Runtime Repository Service where source is the Development (DR) environment and target is the Target (RR or production) environment. (You can use the Runtime Repository Service instead of Incremental Runtime Repository Service.)
   For more information about migration planning, see *Creating a Migration Plan*.
3. Run the migration plan that you created in Step 2.
4. After you complete Steps 1 to 3, you must add the seed data to the target (RR or production) environment.
   For more information, see *Adding Seed Data to the Production Environment*.

Note: In a migration plan, if you have chosen a target (RR or production) environment that is in a Repository Upgraded state, then the source (DR or development) environment must also be in a Repository Upgraded state. Otherwise, you will not be able to migrate.

Adding Seed Data to the Target Environment

After you complete the steps described in *Migrating Repository Changes from Development to Production Environment (Siebel CRM 18.8 or Later)*, you must complete the following procedure to add the seed data to the production (target RR) environment.

To add seed data to the production environment

1. In Siebel CRM, open the Call Center application from the development environment that is upgraded to the Siebel 18.8 Update or later release binary and repository.
2. Navigate to the Site Map and then Administration Application - Business Service Access.
3. In the Business Service applet, query for the Application Migration Utility Service in the Name field.
4. In the Business Service Method applet, verify that it contains the following methods:
   - `CreateManifest`
   - `GenerateWatermark`
   - `GetWatermarkForIncrementalImport`
   - `GetWatermarks`
   - `ReadManifest`
   - `ValidateManifest`
   - `GetWatermarkFrmFile`
5. In Siebel CRM on the target (RR or production) environment that you plan to upgrade to the 18.8 Update or later Siebel Runtime Repository, open the Call Center application. This step assumes that the binary is already upgraded to 18.8 Update or later. If not, upgrade the binary.
6. Navigate to the Site Map and then Administration Application - Business Service Access.
7. Create a new record in the Business Service applet.
8. Enter Application Migration Utility Service for the value in the Name field.
9. In the Access By Responsibility applet, add the appropriate responsibility. The default responsibility is Siebel Administrator.

10. In the Business Service Access Method applet, add all the methods listed in Step 4.
11. Save your changes.
12. Restart the EAI Object Manager component (`EAIObjMgr_<lang>`) to read the newly added data.

Setting the Seed Migration Priority System Preference

When migrating from a development (source or Design Time Repository) to a production (target or Runtime Repository) environment, you should set the Seed Migration Priority system preference as required in the target environment in order to resolve any conflicts that might arise when migrating from the DR to RR environment. Set Seed Migration Priority in the target (RR) environment to one of the following values:

1. **Source.** Precedence is given to the seed data in the DR environment.
2. **Target.** Precedence is given to the seed data in the RR environment.

If Seed Migration Priority is not set at all, then the default behavior is to assume Target as the priority and precedence is given to the seed data in the RR environment.

Note the following:

- If data on the target environment has not been modified and Last Updated Source is set to `dataimp` (seed data indication), then independently of the seed migration priority system preference value, target data will be replaced by the source data if it was modified at source.
- The conflict resolution is based on LOV data (About Record - Last Updated Source value). If an LOV's Last Updated Source value is one of the following, this means that the LOV has been modified and that the target setting will take precedence: `User`, `Object Manager - Default`, or `EIM`.

  After the migration plan is executed, Last Updated Source will default to `User` for all modified LOVs.

Before performing your first Full Runtime Repository migration, you must synchronize Siebel LOV data between development (source) and production (target) environments. Set Seed Migration Priority to Source for the first migration – you can subsequently change this to `"Seed Migration Priority" = Target` as required.

For more information about the Seed Migration Priority system preference, see the topic about modifying LOVs in RR environments in *Siebel Applications Administration Guide* .

Setting the RepVer Column Compatibility System Preference

For DB2 database customers performing a Full Migration where the source environment is upgraded to 21.10 or later but the target environment is not upgraded, the import on the target environment will fail if the physical schema has not been migrated from the source to the target environment (something you would do if you want to manually apply the schema changes outside of RepositoryUpgrade). To avoid this import failure, add the RepVer Column Compatibility system preference to the source environment (if not already added) and set it to False.

```
RepVer Column Compatibility=False
```

Migrating User Preferences

The migration of user preferences is typically supported between releases provided that existing SPF files are placed in the Siebel File System on the upgraded system. For example, if you set your Default Greeting Template to My Template, then this preference will be preserved across releases.

However, note that if something changes between releases, then old preferences may be ignored. For example, if significant changes are made to Siebel Calendar (like they were in Siebel CRM 16.x), then many of the preferences related to Calendar will change as a result, making the old preferences obsolete. Any addition of new preferences is backward compatible.

Note: Prior to Siebel CRM 8.1.x, the migration of user preferences from one major release to another (for example, from 7.8 to 8.1) is not supported. In such cases, it is recommended that you rename or delete your SPF files with each major upgrade and then reset your user preferences after the upgrade completes. New SPF files are generated during the upgrade.

Migrating Configuration Data and Incremental Changes

When migrating configuration data (such as LOVs, runtime events, and data mapper) and incremental changes, note the following:

- LOVs are versioned and may change from one incremental migration to the next.
- Performing a full migration creates a single instance of all repository objects and LOVs with version 0.
- During incremental migrations, new and/or modified LOVs from the DR are created in the RR with new version numbers.
- To seed a new Target environment with all the LOVs and repository objects, the first migration that you perform must always be a full migration.

  Note: After a flatten Workspace operation, the next migration must always be a full migration. For more information about flattening Workspaces, see *Using Siebel Tools* .

- Since LOVs are Workspace-enabled, use the Application Workspace Data Service and Incremental Application Workspace Data Service to migrate LOV data including incremental changes.

  For all other configuration data (bar incremental changes), use the Application Data Service, Application Data Service with Transformation, or Application Deployment Manager Projects to migrate data.

  - For example, to perform a full migration of LOVs using Siebel Migration Application, select the following resource options:
    - Schema Service
    - Runtime Repository Data Service
    - Application Workspace Data Service
  - For example, to perform an incremental migration of LOVs using Siebel Migration Application, select the following resource options:
    - Schema Service
    - Incremental Runtime Repository Data Service
    - Incremental Application Workspace Data Service

Migrated Configuration Data Is Available to the Runtime Repository Without User Re-Login

When configuration is migrated to Runtime Repositories such as in Test/QA/UAT/Production using Incremental Runtime Repository Merge, that configuration is available to logged in users after the poll time that looks for new configuration. The poll time could be fifteen minutes which is the default setting or a different duration if you configured it differently. After the new configuration is loaded the application presents the new configuration when the next navigation occurs such as navigating to a new view.

Note: This functionality is enabled by default but can be disabled by setting the below parameters.

This fuctionality allows:

- Changed objects such as Views, Applets, Business Components, Server Script, Browser script, PickList values.
- New objects such as Views, Controls on Applets, new Applets, Menu Items, Page Tabs, Find Items, Toolbar Items.
- Deleted/Inactivated objects such as Views, Applets, Fields.
- Changed or added Runtime Events.
- Pausing Task/Workflow Processes will resume under old definition.

Note: In-flight processes, such as Workflow Processes or Tasks continues to completion using the version of the process, loaded to start the process. For Example, if a Task or Workflow Process started in Version 1 the running instance of the Task or Workflow Process continues to use Version 1 until it completes. The *next* invocation of that same Task or Workflow Process uses the updated version.

About the Parameters

Poll Time

This Object Manager Parameter sets the poll time that is used to detect the migration of new configuration. In this example, the parameter is set for the Call Center Object Manager, but you would set this for the Object Manager that you are using. The default is 900 seconds, that is 15 minutes.

```
change param ServerSessionLoopSleepTime=900 for comp SCCObjMgr_enu
```

Note: Do not set the `ServerSessionLoopSleepTime` value less than 60.

Related Object Manager Parameters

These two parameters are necessary to enable this functionality and should automatically be set when the Monthly Update is installed in your Runtime Repository environment. These parameters are documented here for completeness but changing these parameters is not supported unless directly advised by Oracle Support. This example is for the Call Center Object Manager.

```
change param ServerSessionBusSvc=CSSWSUtilityService for comp SCCObjMgr_enu

change param ServerSessionBusSvcMethod=GetLatestWSVersion for comp SCCObjMgr_enu
```

Pausing/Resuming This Feature

You would not want to pause this feature as it removes an inconvenience to the users. Without this feature enabled the logged in users will not be able to use the latest configuration, until they log out and login back again. However, in case you find some situation where you want the users to re-login to receive your latest updates, the following parameters can be changed to disable/enable the feature.

To pause this feature and not receive the latest Workspace Configurations, set the following Object Manager Parameter:

```
change param ServerSessionLoopRunning=FALSE for comp SCCObjMgr_enu
```

To resume receiving latest Workspace Configurations set the following Object Manager Parameter:

```
change param ServerSessionLoopRunning=TRUE for comp SCCObjMgr_enu
```

Note: After setting these parameters there is no need to restart the Object Manager. The parameters will take effect immediately.

Component Parameter to Auto-detect Updated Configuration in the Runtime Repository

In the runtime repository user interface, a newly migrated configuration is used automatically when you navigate to a new view if the application detects it within the specified time. The default time is 15 minutes.

This parameter was introduced in the Siebel monthly update version 23.1. But some components of the Siebel application don't “navigate” to a different view, triggering the application to check if a new configuration has migrated. Examples of such technology are Workflow Processes, an EAI processes, or incoming REST requests.

To allow these processes to also pick up the new configuration in your design repository environment, set a parameter in the component that's used to handle the type of request. For example, the EAI Object Manager would be used to handle EAI requests.

This parameter is used to immediately recognize and use a configuration that has been migrated in a design repository environment without waiting for the time out to check.

> **Note:** Because this check for new configuration happens quite often, it's disabled by default to prevent performance degradation. Enable this only for testing.

Component Parameter Details

Parameter Name	Parameter Value
TurnOnVersionDetectionOnDR	**True** - Turns on detection that new configuration has been migrated **False** (Default) - Turns off detection of new configuration

> **Note:** This is only appropriate in your design repository. This Component Parameter has no effect on your runtime repository environment.

Managing Cross Version Migration

Siebel CRM applications in general support *cross version migration*, meaning that you can use Siebel Migration to propagate changes from a Design Repository (DR) environment which has a later binary version than the Runtime Repository (RR) environment. For example, consider a scenario where you have installed the latest Siebel CRM monthly update in the DR environment, but are not planning to install the same monthly update in the RR environment until you have had adequate time to test it – perhaps a few months or more later. In the meantime, you may have repository or other changes that you want to migrate from DR to RR, even though they are on different binary versions.

In this scenario, some types of changes need to be managed carefully – for example:

- At the boundary between Siebel CRM 20.6 and 20.7 when the *Workspace-Enablement of Workflow Processes* feature was introduced. This feature includes changes to Workflow Process-related tables, in particular the RR tables where Workflow Processes are deployed. For a customer on Siebel CRM 20.7 (or later) in the DR and on Siebel CRM 20.6 (or earlier) in the RR, Siebel Migration needs to know what format the target environment is anticipating – is it expecting to receive data from the old deployment table or the new deployment table?
- At the boundary between Siebel CRM 22.6 and 22.7 when the *Workspace-Enablement of Tasks* feature was introduced. This feature includes changes to task flow-related tables, in particular the RR tables where task flows are deployed. For a customer on Siebel CRM 22.7 (or later) in the DR and on Siebel CRM 22.6 (or earlier) in the RR, Siebel Migration needs to know what format the target environment is anticipating – is it expecting to receive data from the old deployment table or the new deployment table?

The answer in both cases: The target environment expects to receive DR data from the source environment, and the data is deployed into the old deployment table in the target using the old business service.

Cross Version - Incremental Migration

For incremental migration of workflow objects, any version after Siebel CRM 20.7 will handle cross version migration automatically:

- When the watermark is generated by the RR environment, it will include the target's binary version.
- When the migration export job is executed in the DR, it will use this information to manage any differences between the two environments.
- If the target is pre-20.7, it will send the Workflow Process data in the older format.
- If the target is version 20.7 or later, it will use the newer format.

For incremental migration of task flow objects, any version after Siebel CRM 22.7 will handle cross version migration automatically:

- When the watermark is generated by the RR environment, it will include the target's binary version.
- When the migration export job is executed in the DR, it will use this information to manage any differences between the two environments.
- If the target is pre-22.7, it will send the task flow data in the older format.
- If the target is version 22.7 or later, it will use the newer format.

Cross Version - Full Migration

For full migration, there is no watermark so there is no way for Siebel Migration to know that the target environment is on a different version. To accommodate this – that is, to let Siebel Migration know that the target environment has an earlier version, add the FullMigCompatibilityMode system preference to the source (DR) environment. Doing this ensures that the expected data is sent to the target (RR) environment.

Note the following about adding and configuring the FullMigCompatibilityMode system preference:

1. This parameter is only required when crossing a release boundary where a change has occurred.
   - In the example provided here for workflow objects, it is only required if the DR is on Siebel CRM 20.7 or later and the RR is on Siebel CRM 20.6 or earlier.
   - In the example provided here for task flow objects, it is only required if the DR is on Siebel CRM 22.7 or later and the RR is on Siebel CRM 22.6 or earlier.
2. Since each target environment's binary version can be different (for example, if your Test environment is on 20.5/22.7 and your Production environment is on 20.3/22.6), you can create multiple "FullMigCompatibilityMode" system preferences using sequential numbers. That is, FullMigCompatibilityMode0, FullMigCompatibilityMode1, FullMigCompatibilityMode2, and so on.
3. For each source Integration Workspace (IWS) from which you will be migrating, you must create a "FullMigCompatibilityModeX" system preference that contains the name of the source IWS (or MAIN):
   - **System Preference Name:** *FullMigCompatibilityModeX*, where X is a consecutive number starting from zero for each IWS that will be migrated across a known release boundary.
   - **System Preference Value:** *Workflow, Task* or some other object type that changes across a given version boundary. Set "Value" to one of the Values shown in the following (second) table.
   - **Description:** *<The name of the IWS (or MAIN)>* for which the compatibility mode is required.

Example configurations for FullMigCompatibilityMode are shown in the following table.

System Preference Name	System Preference Value	Description
FullMigCompatibilityMode0	Workflow	MAIN
FullMigCompatibilityMode1	Task	MAIN
FullMigCompatibilityMode2	Workflow	Int_June_202x
FullMigCompatibilityMode3	Task	Int_June_202x
FullMigCompatibilityMode4	Workflow	Int_July_202x
FullMigCompatibilityMode5	Task	Int_July_202x

The "System Preference Value" refers to the type of object that is of concern to cross a boundary and should be set for example as shown in the following table.

Source Version >=	Target Version <=	Value
20.7	20.6	Workflow
22.7	22.6	Task

Note: Only workflows and task flows are affected (during cross version migration) at this time. As more objects become Workspace-enabled, new boundaries will be determined.

CleanUpTaskDR Utility

In the case of a full migration where the source is being upgraded to 22.7 or later and the RR environment has task DR data, any import of task DR records will fail due to a *duplicate record error*. For the import to run without error on a lower versioned environment, task DR data must be deleted. To delete task DR data, you must run the CleanUpTaskDR utility located in the `.../ses/siebsrvr/bin` folder.

To run the CleanUpTaskDR utility

1. Copy the CleanUPTaskDR utility from `SiebelBuild/ses/siebsrvr/bin` on the source environment to the same location on the target environment.
2. Run the CleanUpTaskDR utility from `SiebelBuild/ses/siebsrvr/bin` on the target environment.

   The syntax for running the CleanUpTaskDR utility is as follows:

   ```
   SiebelBuild/ses/siebsrvr/bin> CleanupTaskDR -t <TBLO> -u <TBLO User> -p <TBLO User Password>
   ```

```
-o <ODBC Data Source> -s <Siebsrvr Installation Path>
```

For example:

```
C:\2022_06\ses\siebsrvr\bin CleanupTaskDR -t dbo -u SADMIN -p MSSQL
-o Siebel_DSN -s C:\2022_06\ses\siebsrvr
```

While running this command, the following message appears:

```
C:\2022_06\ses\siebsrvr\bin> Deleting data from Task Related Tables
```

The following table describes the arguments that you can use to run the CleanUpTaskDR utility.

Argument	Description	Example
`-t`	Table Owner	`dbo`
`-u`	Table Owner User	`SADMIN`
`-p`	Table Owner Password	`********`
`-o`	ODBC Data Source	`siebel_DSN`
`-l`	Log File Name (Optional)	`CleanupTaskDR_timestamp.log` (Default)
`-s`	Siebsrvr Installation Path	`C:\2022_07\ses\siebsrvr`

Troubleshooting Data Migration Using Siebel Migration

Important issues to note when using Siebel Migration to migrate data include the following:

- If you opt to use the Migration Package Location when creating a Siebel Migration Profile, then make sure that the process running the Siebel Object Manager has read-write access to the network file share path for the Migration Package Location. For more information, see *Roadmap for Planning a Migration with Siebel Migration*.

24 REST API References for Migration Services

REST API References for Migration Services

This chapter provides examples for using REST API to discover migration services. It includes the following topics:

- *Using REST API with the Migration Schema Service*
- *Using REST API with the Migration Application Data Service*
- *Using REST API with the Migration Data Service with Transformation Service*
- *Using REST API with the Migration Incremental Runtime Repository Data Service*
- *Using REST API with the Migration Runtime Repository Data Service*
- *Using REST API with the Migration Incremental Application Workspace Data Service*
- *Using REST API with Migration Application Workspace Data Service*
- *Using REST API with the Migration File Prepare and Deploy Service*
- *Using REST API with the Siebel Migration Application*

Using REST API with the Migration Schema Service

The Migration Schema Service migrates the physical Siebel schema from the source environment to the target environment. The following table shows the methods supported by the Migration Schema Service.

Method	**Definition**
Export	Method used to export a schema for a migration.
Import	Method used to import a schema for a migration.
GetWatermark	Method used to get a watermark for a migration.
IsSchemaChanged	Method used to check if a schema has changed for a migration.
GetStatus	Method used to get the status of a migration.

This topic includes the following topics:

- *Exporting with the Migration Schema Service*
- *Getting Status with the Migration Schema Service*

- *Importing with the Migration Schema Service*
- *Verifying If a Schema Changed with the Migration Schema Service*
- *Getting a Watermark with the Migration Schema Service*

Exporting with the Migration Schema Service

You can export a Migration Schema resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to export a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Schema Service/Export`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameter:
    - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
- Request body:

```
{
 "body":{
 "migrationid": "<Migration Id value>
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
    - trackingid: String that contains the tracking identification value.
- Response body:

```
{
 "trackingid": "<tracking ID value>"
}
```

Getting Status with the Migration Schema Service

You can get status for a Migration Schema resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to get status for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Schema Service/GetStatus`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic

- Request parameters:
  - trackingid: Contains the tracking identification value.
  - getlog: If the value of this parameter is true, the log content of the resource along with the status is returned in the response. If the value is false, only the status is returned in the response.
  - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
- Request body:

```
{"body":{
 "trackingid": "<tracking Id value>",
 "getlog": "true"
 "migrationid": "<Migration Id value>
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - status: Returns a value for the status of the request:
    - running: Indicates that the resource is running.
    - success: Indicates that the request was completed successfully.
    - error: Indicates that the request failed and the error parameter is populated with an error message.
  - error: Returns an error message if an error is encountered.
- Response body:

```
{
 "status": "success",
 "error": ""
}
```

Importing with the Migration Schema Service

You can import a Migration Schema resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to import a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Schema Service/Import`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
  - filename: Use the Tracking Id value that is present in the response from *Exporting with the Migration Schema Service*.
  - username: Use the user name parameter to enter your database user name.

    - password: Use the password parameter to enter your database password, which must be in Base64 encoded format.
    - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.

- Request body:

```
{"body":{
 "filename": "<Tracking Id value>",
 "username": "<db username>",
 "password": "<base64 encoded database password>"
 "migrationid": "<Migration Id value>
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
    - trackingid: Returns the tracking identification value.
- Response body:

```
{
 "trackingid": "<tracking id value>"
}
```

Verifying If a Schema Changed with the Migration Schema Service

You can check if a schema for a Migration Schema resource has changed by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to check if a schema has changed for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Schema Service/IsSchemaChanged`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameter:
    - watermark: Use the watermark parameter to enter the watermark value.
- Request body:

```
{ "body":
 {
 "watermark": "<watermark value>"
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - isschemachanged: Returns the value Y or N. Y indicated that the schema has changed. N indicates that the schema has not changed.
- Response body:

```
{
 "isschemachanged": "Y"
}
```

Getting a Watermark with the Migration Schema Service

You can get a watermark for a Migration Schema resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to get a watermark for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Schema Service/GetWatermark`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request body:

```
{
 "body":{}
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - watermark: Returns the watermark value.
- Response body:

```
{
 "watermark": "<watermark value>"
}
```

Using REST API with the Migration Application Data Service

The Migration Application Data Service migrates the data from the source environment to the target environment based on the tables listed in datamig.inp file on the source environment. The following table shows the methods supported by the Migration Application Data Service.

Method	Definition
Export	Method used to export the application data schema for a migration. For more information, see *Exporting with the Migration Application Data Service*.
Import	Method used to import a schema for a migration. For more information, see *Importing with the Migration Application Data Service*.
GetStatus	Method used to get the status of a migration. For more information, see *Getting Status with the Migration Application Data Service*.

Exporting with the Migration Application Data Service

You can export a Migration Application Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to export a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Application Data Service/Export`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
  - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
- Request body:

```
{
 "body":{

 "migrationid": "<Migration Id value>
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200

- Content-Type: application/json
- Response parameters:
    - trackingid: Returns the tracking identification value.
- Response body:

```
{
 "trackingid": "<tracking id value>"
}
```

Getting Status with the Migration Application Data Service

You can get status for a Migration Application Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to get status for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Application Data Service/GetStatus`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
    - trackingid: Contains the tracking identification value.
    - getlog: If the value of this parameter is true, the log content of the resource along with the status is returned in the response. If the value is false, only the status is returned in the response.
    - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
- Request body:

```
{
 "body":
 {
 "trackingid": "<tracking id value>",
 "migrationid": "<Migration Id value>
 "getlog": "true"
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
    - status: Returns a value for the status of the request:
        - running: Indicates that the resource is running.
        - success: Indicates that the request was completed successfully.
        - error: Indicates that the request failed and the error parameter is populated with an error message.

  - error: Returns an error message if an error is encountered.

- Response body:

```
{
 "status": "success",
 "error": ""
}
```

Importing with the Migration Application Data Service

You can import a Migration Application Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to import a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Application Data Service/Import`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
  - filename: Use the Tracking Id value that is present in the response from *Exporting with the Migration Application Data Service*.
  - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
- Request body:

```
{
 "body":
 {
 "filename": "<Tracking Id value>"
 "migrationid": "<Migration Id value>
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - trackingid: Returns the tracking identification value.
- Response body:

```
{
 "trackingid": "<tracking id value>"
}
```

Using REST API with the Migration Data Service with Transformation Service

The Migration Data Service with Transformation service migrates data from the source environment to the target environment. based on the tables listed in the datamig.inp on the source environment.

While exporting the data, this service uses the rule defined in the datamig.rul file and performs the transformation. The transformed data will be migrated to the target environment.

The following table shows the methods supported by the Migration Data Service with Transformation Service.

Method	Definition
Export	Method used to export the application data for a migration. For more information, see *Exporting with the Migration Application Data Service With Transformation*.
Import	Method used to import the application data for a migration. For more information, see *Importing with the Migration Application Data Service With Transformation*.
GetStatus	Method used to get the status of a migration. For more information, see *Getting Status with the Migration Application Data Service With Transformation*.

Exporting with the Migration Application Data Service With Transformation

You can export a Migration Application Data Service With Transformation resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to export a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Application Data Service With Transformation/Export`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameter:
    - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
- Request body:

```
{
 "body":{
 "migrationid": "<Migration Id value>
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - trackingid: Returns the tracking identification value.
- Response body:

```
{
"trackingid": "<tracking id value>"
}
```

Getting Status with the Migration Application Data Service With Transformation

You can get status for a Migration Application Data Service With Transformation resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to get status for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Application Data Service With Transformation/GetStatus`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
  - trackingid: Contains the tracking identification value.
  - getlog: If the value of this parameter is true, the log content of the resource along with the status is returned in the response. If the value is false, only the status is returned in the response.
  - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
- Request body:

```
{
 "body":
 {
 "trackingid": "<tracking id value>",
 "migrationid": "<Migration Id value>
 "getlog": "true"
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json

- Response parameters:
    - status: Returns a value for the status of the request:
        - running: Indicates that the resource is running.
        - success: Indicates that the request was completed successfully.
        - error: Indicates that the request failed and the error parameter is populated with an error message.
    - error: Returns an error message if an error is encountered.
- Response body:

```
{
 "status": "success",
 "error": ""
}
```

Importing with the Migration Application Data Service With Transformation

You can import a Migration Application Data Service With Transformation resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to import a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Application Data Service With Transformation/Import`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
    - filename: Use the Tracking Id value that is present in the response from *Exporting with the Migration Application Data Service With Transformation*.
    - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
- Request body:

```
{
 "body":
 {
 "filename": "<Tracking Id value>"
 "migrationid": "<Migration Id>
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:

  - trackingid: Returns the tracking identification value.

- Response body:

```
{
 "trackingid": "<tracking id value>"
}
```

Using REST API with the Migration Incremental Runtime Repository Data Service

The Migration Incremental Runtime Repository Data Service identifies the version of the repository data that was previously migrated. If you select the latest migration version, then this service takes the changes from the previous migration version and latest version and migrates the data to the target environment. If you do not make any selections, the service only considers the latest migration version and migrates the data to the target environment.

The following table shows the methods supported by the Migration Incremental Runtime Repository Data Service.

Migration Incremental Runtime Repository Data Service Supported Methods

Method	Definition
Export	Method used to export the incremental Runtime Repository changes for a migration. For more information, see *Exporting with the Migration Incremental Runtime Repository Data Service*.
Import	Method used to import the incremental Runtime Repository changes for a migration. For more information, see *Importing with the Migration Incremental Runtime Repository Data Service*.
GetWatermark	Method used to get a watermark for a migration. For more information, see *Getting a Watermark with the Migration Incremental Runtime Repository Data Service*.
DBCheck	Method used to check a database has for a migration. For more information, see *Checking a Database with the Migration Incremental Runtime Repository Data Service*.
GetStatus	Method used to get the status of a migration. For more information, see *Getting Status with the Migration Incremental Runtime Repository Data Service*.

Getting a Watermark with the Migration Incremental Runtime Repository Data Service

You can get a watermark for a Migration Incremental Runtime Repository Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to get a watermark for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Incremental Runtime Repository Data Service/GetWatermark`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request body:

```
{
 "body":{}
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - watermark: Returns the watermark value.
- Response body:

```
{
 "watermark": "<watermark value>"
}
```

Exporting with the Migration Incremental Runtime Repository Data Service

You can export a Migration Incremental Runtime Repository Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to export a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Incremental Runtime Repository Data Service/Export`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
  - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
  - watermark: Use the watermark parameter to include the watermark value.
- Request body:

```
{
 "body":
```

```
 {
 "migrationid", "<Migration Id value>
 "watermark": "<watermark value>"
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - trackingid: Returns the tracking identification value.
- Response body:

```
{
"trackingid": "<tracking id value>"
}
```

Getting Status with the Migration Incremental Runtime Repository Data Service

You can get status for a Migration Incremental Runtime Repository Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to get status for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/ Migration Incremental Runtime Repository Data Service/ GetStatus`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
  - trackingid: Contains the tracking identification value.
  - getlog: If the value of this parameter is true, the log content of the resource along with the status is returned in the response. If the value is false, only the status is returned in the response.
  - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
- Request body:

```
{
"body":
 {
 "trackingid": "<tracking id value>",
 "migrationid", "<Migration Id value>"
 "getlog": "true"
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - status: Returns a value for the status of the request:
    - running: Indicates that the resource is running.
    - success: Indicates that the request was completed successfully.
    - error: Indicates that the request failed and the error parameter is populated with an error message.
  - error: Returns an error message if an error is encountered.
- Response body:

```
{
 "status": "success",
 "error": ""
}
```

Importing with the Migration Incremental Runtime Repository Data Service

You can import a Migration Incremental Runtime Repository Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to import a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/ Migration Incremental Runtime Repository Data Service/ Import`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
  - filename: Use the Tracking Id value that is present in the response from *Exporting with the Migration Incremental Runtime Repository Data Service*.
  - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
- Request body:

```
{
"body":
 {
 "filename": "<Tracking Id value>"
 "migrationid", "<Migration Id value>"
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
    - trackingid: Returns the tracking identification value.
- Response body:

```
{
"trackingid": "<tracking id value>"
}
```

Checking a Database with the Migration Incremental Runtime Repository Data Service

You can check a database for a Migration Incremental Runtime Repository Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to check a database for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/ Migration Incremental Runtime Repository Data Service/ DBCheck`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request body:

```
{
 "body":{}
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
    - trackingid: Returns the tracking identification value.
- Response body:

```
{
"trackingid": "<tracking id value>"
}
```

Using REST API with the Migration Runtime Repository Data Service

The Migration Runtime Repository Data Service migrates only the Runtime Repository from the source environment to the target environment. The following table shows the methods supported by the Migration Runtime Repository Data Service.

Method	Definition
Export	Method used to export the Runtime Repository for a migration. For more information, see *Exporting with the Migration Runtime Repository Data Service*.
Import	Method used to import the Runtime Repository for a migration. For more information, see *Importing with the Migration Runtime Repository Data Service*.
GetWatermark	Method used to get a watermark for a migration. For more information, see *Getting a Watermark with the Migration Runtime Repository Data Service*.
DBCheck	Method used to check a database for a migration. For more information, see *Checking a Database with the Migration Runtime Repository Data Service*.
GetStatus	Method used to get the status of a migration. For more information, see *Getting Status with the Migration Runtime Repository Data Service*.
GetRRInfo	Method used to get Runtime Repository information. For more information, see *Getting Runtime Repository Information with the Migration Runtime Repository Data Service*.

Getting Runtime Repository Information with the Migration Runtime Repository Data Service

You can get Runtime Repository information for a Migration Runtime Repository Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to get Runtime Repository information for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Runtime Repository Data Service/GetRRInfo`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request body:

```
{
 "body":{}
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - workspace: Returns the name of the Workspace branch, the latest version of the Workspace branch, and Workspace languages. If there are multiple Workspace branches in the Siebel environment, then the branch name and its latest version will be separated by a comma in the response. The response lists all the Siebel environment languages. If there is only one language, then only one language is listed in the response.
- Response body:

```
{
 "workspace":
 {
 "Branch Name": "Last Version Number of the Branch"
 },
 "languages":
 {...
 }
}
```

Getting a Watermark with the Migration Runtime Repository Data Service

You can get a watermark for a Migration Runtime Repository Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to get a watermark for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Runtime Repository Data Service/GetWaterMark`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
  - workspace: Use the Workspace parameter to enter the name of the Workspace branch in the REST API request.
  - version: Use the version parameter to enter the version number of the Workspace branch. The value is 0 to the latest version of the specified Workspace branch.
- Request body:

```
{
 "body":
 {
 "workspace":"<Workspace branch name>",
 "version":"<version>",
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - watermark: Returns the watermark value.
- Response body:

```
{
 "watermark": "<Watermark value>"
}
```

Exporting with the Migration Runtime Repository Data Service

You can export a Migration Runtime Repository Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to export a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Runtime Repository Data Service/Export`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
  - watermark: Use the watermark parameter to include the name of the watermark in the REST API request.
  - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
- Request body:

```
{
 "body":
 {
 "watermark": "<Watermark value>"
 "migrationid", "<Migration Id value>"
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - trackingid: Returns the tracking identification value.
- Response body:

```
{
  "trackingid": "<tracking id value>"
}
```

Getting Status with the Migration Runtime Repository Data Service

You can get status for a Migration Runtime Repository Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to get status for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Runtime Repository Data Service/GetStatus`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
    - trackingid: Contains the tracking identification value.
    - getlog: If the value of this parameter is true, the log content of the resource along with the status is returned in the response. If the value is false, only the status is returned in the response.
    - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
- Request body:

```
{
 "body":
 {
 "trackingid":"<tracking id value>",
 "migrationid", "<Migration Id value>"
 "getlog":"TRUE"
}
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
    - status: Returns a value for the status of the request:
        - running: Indicates that the resource is running.
        - success: Indicates that the request was completed successfully.
        - error: Indicates that the request failed and the error parameter is populated with an error message.
    - error: Returns an error message if an error is encountered.
    - log: Returns log file content if the getlog parameter value is set to TRUE.
- Response body:

```
{
  "status": "success",
  "error": "",
  "log": <log file content>
}
```

Importing with the Migration Runtime Repository Data Service

You can import a Migration Runtime Repository Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to import a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Runtime Repository Data Service/Import`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
    - filename: Use the Tracking Id value that is present in the response from *Exporting with the Migration Runtime Repository Data Service*.
    - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
- Request body:

```
{
 "body":
 {
 "filename": "<Tracking Id value>",
 "migrationid", "<Migration Id value>"
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
    - trackingid: String that contains the tracking identification value.
- Response body:

```
{
 "trackingid": "<tracking id value>"
}
```

Checking a Database with the Migration Runtime Repository Data Service

You can check a database for a Migration Runtime Repository Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to check a database for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Runtime Repository Data Service/DBCheck`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request body:

```
{
 "body":{}
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - trackingid: Returns the tracking identification value.
- Response body:

```
{
"trackingid": "<tracking id value>"
}
```

Using REST API with the Migration Incremental Application Workspace Data Service

The Migration Incremental Application Workspace Data Service identifies the version that was previously migrated. This service takes all the changes from the previously migrated version to the latest version and migrates them to the target environment. The following table shows the methods supported by the Migration Incremental Application Workplace Data Service. For information on how to invalidate seed caches, see *Invalidating Seed Caches with the Migration Incremental Application Workspace Data service*.

Method	Definition
Export	Method used to export the Workspace data for a migration. For more information, see *Exporting with the Migration Incremental Application Workspace Data Service*.

Method	Definition
Import	Method used to import the Workspace data for a migration. For more information, see *Importing with the Migration Incremental Application Workspace Data Service*.
GetWatermark	Method used to get a watermark for a migration. For more information, see *Getting a Watermark with the Migration Incremental Application Workspace Data Service*.
GetStatus	Method used to get the status of a migration. For more information, see *Getting Status with the Migration Incremental Application Workspace Data Service*.

Getting Status with the Migration Incremental Application Workspace Data Service

You can get status for a Migration Incremental Application Workspace Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to get status for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Incremental Application Workspace Data Service/GetStatus`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
  - trackingid: Contains the tracking identification value.
  - getlog: If the value of this parameter is true, the log content of the resource along with the status is returned in the response. If the value is false, only the status is returned in the response.
  - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
- Request body:

```
{
 "body":
 {
 "trackingid":"<tracking id value>",
 "migrationid", "<Migration Id value>"
 "getlog":"TRUE"
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json

- Response parameters:
  - status: Returns a value for the status of the request:
    - running: Indicates that the resource is running.
    - success: Indicates that the request was completed successfully.
    - error: Indicates that the request failed and the error parameter is populated with an error message.
  - error: Returns an error message if an error is encountered.
  - log: Return log file content if the getlog parameter is set to TRUE.
- Response body:

```
{
"status": "success",
"error": "",
"log": log file content
}
```

Getting a Watermark with the Migration Incremental Application Workspace Data Service

You can get a watermark for a Migration Incremental Application Workspace Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to get a watermark for a resource:

- URI: **`http://<host>:<port>/siebel/v1.0/service/Migration Incremental Application Workspace Data Service/GetWaterMark`**
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request body:

```
{
 "body":{}
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - watermark: Returns the watermark value.
- Response body:

```
{
 "watermark": "watermark value"
}
```

Exporting with the Migration Incremental Application Workspace Data Service

You can export a Migration Incremental Application Workspace Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to export a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Incremental Application Workspace Data Service/Export`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
  - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
  - watermark: Use the watermark parameter to include the watermark value in the REST API request.
- Request body:

```
{
 "body":
 {
 "migrationid", "<Migration Id value>"
 "watermark": "<watermark value>"
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - trackingid: Returns the tracking identification value.
- Response body:

```
{
"trackingid": "<tracking id value>"
}
```

Importing with the Migration Incremental Application Workspace Data Service

You can import a Migration Incremental Application Workspace Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to import a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Incremental Application Workspace Data Service/Import`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
  - filename: Use the Tracking Id value that is present in the response from *Exporting with the Migration Incremental Application Workspace Data Service*.
  - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
- Request body:

```
{
 "body":
 {
 "filename": "<Tracking Id value>"
 "migrationid", "<Migration Id value>"
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - trackingid: Returns the tracking identification value.
- Response body:

```
{
"trackingid": "<tracking id value>"
}
```

Invalidating Seed Caches with the Migration Incremental Application Workspace Data service

You can invalidate seed caches for a Migration Incremental Application Workspace Data service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to invalidate seed caches for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Incremental Application Workspace Data service/InvalidateSeedCaches`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic

- Request body:

```
{
 "body":{}
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - status: Returns a value for the status of the request:
    - running: Indicates that the resource is running.
    - success: Indicates that the request was completed successfully.
    - error: Indicates that the request failed and the error parameter is populated with an error message.
- Response body:

```
{
 "status": "success"
}
```

Using REST API with Migration Application Workspace Data Service

The Migration Application Workspace Data Service migrates the seed records from the source environment to the target environment. The following table shows the methods supported by the Migration Application Workspace Data Service. For information on how to invalidate seed caches, see *Invalidating the Seed Caches with the Migration Application Workspace Data Service*.

Method	Description
GetSeedCopyWatermark	Method used to get the watermark that is needed for the SeedCopyExport method for a migration. For more information, see *Getting a Seed Copy Watermark with the Migration Application Workspace Data Service*.
GetFullSeedWatermark	Method used to get a watermark that is needed for the FullSeedExport for a migration. For more information, see *Getting the Full Seed Watermark with the Migration Application Workspace Data Service*.
SeedCopyExport	Method used to export the Workspace data (LOV) from the Siebel Repository in the target environment. For more information, see *Getting a Seed Copy Export with the Migration Application Workspace Data Service*.
SeedCopyImport	Method used to import the Workspace data (LOV), that was exported using the SeedCopyExport method, into the Migrated Repository in the target environment. This helps move the additional LOV's that are present in the Siebel Repository to the Migrated Repository. For more information, see *Getting a Seed Copy Import with the Migration Application Workspace Data Service*.

Method	Description
FullSeedExport	Method used to export the Workspace data (LOV) from a chosen Integration Workspace in the source environment. For more information, see *Getting the Full Seed Export with the Migration Application Workspace Data Service*.
FullSeedImport	Method used to import the Workspace data (LOV), that was exported using the FullSeedExport method, into the Migrated Repository in the target environment. For more information, see *Getting the Full Seed Import with the Migration Application Workspace Data Service*.
GetStatus	Method used to get the status of a migration. For more information, see *Getting Status with the Migration Application Workspace Data Service*.

Getting a Seed Copy Watermark with the Migration Application Workspace Data Service

You can get a seed copy watermark for a Migration Application Workspace Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to get a seed copy watermark for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Application Workspace Data Service/GetSeedCopyWatermark`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request body:

```
{
 "body":{}
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - watermark: Returns the watermark value.
- Response body:

```
{
"watermark": "<Watermark value>"
}
```

Getting the Full Seed Watermark with the Migration Application Workspace Data Service

You can get the full seed watermark for a Migration Application Workspace Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to get the full seed watermark for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Application Workspace Data Service/GetFullSeedWatermark`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request body:

```
{
 "body":{}
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - watermark: Returns the watermark value.
- Response body:

```
{
"watermark": "<Watermark value>"
}
```

Getting Status with the Migration Application Workspace Data Service

You can get status for a Migration Application Workspace Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to get status for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Application Workspace Data Service/GetStatus`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:

    - trackingid: Contains the tracking identification value.
    - getlog: If the value of this parameter is true, the log content of the resource along with the status is returned in the response. If the value is false, only the status is returned in the response.
    - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
- Request body:

```
{
 body":
 {
 "trackingid":"<tracking id value>"
 "migrationid", "<Migration Id value>"
 "getlog":"TRUE"
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
    - status: Returns a value for the status of the request:
        - running: Indicates that the resource is running.
        - success: Indicates that the request was completed successfully.
        - error: Indicates that the request failed and the error parameter is populated with an error message.
    - error: Returns an error message if an error is encountered.
    - log: Returns log file content if the getlog parameter value is set to TRUE.
- Response body:

```
{
"status": "success",
"error": "",
"log": "log file content
}
```

Getting a Seed Copy Export with the Migration Application Workspace Data Service

You can get a seed copy export for a Migration Application Workspace Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to get a seed copy export for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Application Workspace Data Service/SeedCopyExport`
- HTTP Method: POST
- Content-Type: application/json

- Authorization: Basic
- Request parameters:
    - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
    - watermark: Use the watermark parameter to include the watermark value in the REST API request.
- Request body:

```
{
 "body":
 {
 "migrationid", "<Migration Id value>"
 "watermark": "<watermark value>"
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
    - trackingid: Returns the tracking identification value.
- Response body:

```
{
 "trackingid": "<tracking id value>"
}
```

Getting a Seed Copy Import with the Migration Application Workspace Data Service

You can get a seed copy import for a Migration Application Workspace Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to get a seed copy import for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Application Workspace Data Service/SeedCopyImport`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
    - filename: Use the Tracking Id value that is present in the response from *Getting a Seed Copy Export with the Migration Application Workspace Data Service*.
    - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
- Request body:

```
{
 "body":
 {
 "filename": "<Tracking Id value>"
 "migrationid", "<Migration Id value>"
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - trackingid: Returns the tracking identification value.
- Response body:

```
{
 "trackingid": "<tracking id value>"
}
```

Getting the Full Seed Export with the Migration Application Workspace Data Service

You can get a full seed export for a Migration Application Workspace Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to get a full seed export for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Application Workspace Data Service/FullSeedExport`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
  - watermark: Use the watermark parameter to include the watermark value in the REST API request.
  - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
- Request body:

```
{
 "body":
 {
 "migrationid", "<Migration Id value>"
 "watermark": "<watermark value>"
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - trackingid: Returns the tracking identification value.
- Response body:

```
{
 "trackingid": "<tracking id value>"
}
```

Getting the Full Seed Import with the Migration Application Workspace Data Service

You can get full seed import for a Migration Application Workspace Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to get full seed import for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Application Workspace Data Service/FullSeedImport`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
  - filename: Use the Tracking Id value that is present in the response from *Getting the Full Seed Export with the Migration Application Workspace Data Service*.
  - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
- Request body:

```
{
"body":
 {
 "filename":"<Tracking Id value>"
 "migrationid", "<Migration Id value>"
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - trackingid: Returns the tracking identification value.
- Response body:

```
{
 "trackingid": "<tracking id value>"
}
```

Invalidating the Seed Caches with the Migration Application Workspace Data Service

You can invalidate seed caches for a Migration Application Workspace Data Service resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to invalidate seed caches for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/service/Migration Application Workspace Data Service/InvalidateSeedCaches`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request body:

```
{
 "body":{}
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
- Response body:

```
{
"status": "success"
}
```

Using REST API with the Migration File Prepare and Deploy Service

The Migration File Prepare and Deploy Service identifies all the new and modified files and migrates those files from the source environment to the target environment. The following table shows the methods supported by the Migration File Prepare and Deploy Service.

Method	Definition
Export	Method used to export modified files. For more information, see *Exporting with the Migration File Prepare and Deploy Service*.
Import	Method used to import modified files. For more information, see *Importing with the Migration File Prepare and Deploy Service*.
generateWatermark	Method used to generate a watermark for modified files. For more information, see *Generating a Watermark with the Migration File Prepare and Deploy Service*.
readwatermarkfile	Method used to read the watermark for modified files. For more information, see *Reading the Watermark with the Migration File Prepare and Deploy Service*.
writewatermarkfile	Method used to write the watermark for modified files. For more information, see *Writing the Watermark with the Migration File Prepare and Deploy Service*.
GetStatus	Method used to get the status of modified files. For more information, see *Getting Status with the Migration File Prepare and Deploy Service*.

Exporting with the Migration File Prepare and Deploy Service

You can export a File Prepare and Deploy resource by sending an HTTP POST request to the (source) repository resource's URI.

The following details are for a request to export a resource (that is, the *source* files for a File Prepare and Deploy resource):

- URI: `http://<host>:<port>/siebel/v1.0/MigrationFilePrepareAndDeploy/Export`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
  - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
- Request body:

```
{
 "body":{
 "migrationid": "<Migration Id Value>",
 "watermarkFile": "<Watermark File Name>"
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:

- trackingid: Returns the tracking identification value.

- Response body:

```
{
 "trackingid": "<tracking ID value>"
}
```

Getting Status with the Migration File Prepare and Deploy Service

You can get (the API Export or Import) status for a Migration File Prepare and Deploy resource by sending an HTTP POST request to the repository resource's URI.

The following details are for a request to get status for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/MigrationFilePrepareAndDeploy/getstatus`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
  - trackingid: Contains the tracking identification value.
  - getlog: If the value of this parameter is true, the log content of the resource along with the status is returned in the response. If the value is false, only the status is returned in the response.
  - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
- Request body:

```
{"body":{
 "trackingid": "<Tracking Id value>",
 "migrationid": "<Migration Id value>",
 "getlog": "true"
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - status: Returns a value for the status of the request:
    - running: Indicates that the resource is running.
    - success: Indicates that the request was completed successfully.
    - error: Indicates that the request failed and the error parameter is populated with an error message.
  - error: Returns an error message if an error is encountered.
- Response body:

```
{
 "status": "success",
 "error": ""
}
```

Importing with the Migration File Prepare and Deploy Service

You can import a Migration File Prepare and Deploy resource by sending an HTTP POST request to the (target) repository resource's URI.

The following details are for a request to import a resource:

- URI: `http://<host>:<port>/siebel/v1.0/MigrationFilePrepareAndDeploy/Import`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
  - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
- Request body:

```
{"body":{
 "migrationid": "<Migration Id value>
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - trackingid: Returns the tracking identification value.
- Response body:

```
{
 "trackingid": "<tracking id value>"
}
```

Generating a Watermark with the Migration File Prepare and Deploy Service

You can generate a watermark for a Migration File Prepare and Deploy resource by sending an HTTP POST request to the (target) repository resource's URI.

The following details are for a request to generate a watermark for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/MigrationFilePrepareAndDeploy/generateWatermark`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
  - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
  - filename: File name for the watermark.
- Request body:

```
{"body":{
 "migrationid": "<Migration Id value>",
 "Filename": "<File Name>"
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - trackingid: Returns the tracking identification value.
- Response body:

```
{
 "tracking id": "<tracking id value>"
}
```

Reading the Watermark with the Migration File Prepare and Deploy Service

You can read the watermark for a Migration File Prepare and Deploy resource by sending an HTTP POST request to the (target) repository resource's URI.

The following details are for a request to read the watermark for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/MigrationFilePrepareAndDeploy/readwatermarkfile`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
  - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.

- filename: Name of the watermark file.

- Request body:

```
{"body":{
 "migrationid": "<Migration Id value>",
 "Filename": "<File Name>"
 }
}
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - trackingid: Returns the tracking identification value.
  - watermarkContents: Returns the watermark.
- Response body:

```
{
 "tracking id": "<tracking id value>"
 "watermarkContents": "<Watermark contents>"
}
```

Writing the Watermark with the Migration File Prepare and Deploy Service

You can write the watermark received from the target by sending an HTTP POST request to the (source) repository resource's URI.

The following details are for a request to write the watermark (received from the target) for a resource:

- URI: `http://<host>:<port>/siebel/v1.0/MigrationFilePrepareAndDeploy/writewatermarkfile`
- HTTP Method: POST
- Content-Type: application/json
- Authorization: Basic
- Request parameters:
  - migrationid: Use the migrationid parameter to include the migration identification value in the REST API request.
  - filename: File name for the watermark.
  - watermarkContents: Watermark contents received from the target.
- Request body:

```
{"body":{
 "migrationid": "<Migration Id value>",
 "Filename": "<File Name>",
 "watermarkContents": "<Watermark contents>"
```

```
        }
       }
```

The following are the details for the response to a successful request:

- HTTP Code: 200
- Content-Type: application/json
- Response parameters:
  - trackingid: Returns the tracking identification value.
- Response body:

```
{
 "tracking id": "<tracking id value>"
}
```

> **Note:** If you generate the watermark in the target environment and you then want to transfer the Watermark Content to the source environment, then use the ReadWatermark and Writewatermark APIs as detailed in this topic. Otherwise, you must manually copy the Watermark from target to source, and you must do this before performing the export on source.

Using REST API with Siebel Migration Application

You can use REST APIs both before and after the repository upgrade is done. Use the asynchronous migration feature only if both source (development) and target (production) environments are in a Repository Upgraded state. If either (source or target) environment is not in a Repository Upgraded state, then the migration execution will be performed in synchronous mode.

You can use REST API with Siebel Migration Application to do the following:

- Work with connections. You can use REST APIs to create connections, update your existing connection, get information about connections, generate a watermark for your connection, and delete your connection.
- Work with migration plans. You can use REST APIs to create migration plans, update migration plans, get information about migration plans, and delete migration plans.
- Execute migration plans. You can use REST APIs to execute migration plans and get status about your running migration plans by plan name, resource name, and operation. You can also get the log file based on an operation for a running migration plan.
- Get history information about migration plans. You can use REST APIs to get history information about your migration plans by ID, resource name, plan name, and operation. You can also get the log file based on an operation for a particular history record.

This topic contains the following topics:

Using REST API to Configure Siebel Migration Application Connections

You can use the Siebel Migration REST API to configure connections for your migration plan. For more information, see the following subtopics:

- *Getting All Connections*
- *Getting a Connection by Name*
- *Creating a New Connection*
- *Updating the Connection*
- *Refreshing a Connection*
- *Creating a Watermark for a Connection*
- *Deleting a Connection*

Getting All Connections

You can get all connections for your migration by sending an HTTP GET request to the Siebel Migration Application.

The following details are for a request to get all connections for a migration:

- **URI:** `https://{hostname}:{port}/siebel/v1.0/migration/connection`
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic

The following are the details for the response to a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response Body:**

```
[
 {
 "id":"88-1V5WJZ",
 "name":"Dev",
 "restEndpoint":"https://{hostname}:{port}/siebel/v1.0",
 "isFavourite":"false",
 "schemaUser": "",
 "tableSpaceData": "",
 "tableSpaceIndex": "",
 "tableSpacePage16K": "",
 "tableSpacePage32K": "",
 "isUnicodeDatabase": "true",
 "resources":[
 {
 "id":"88-1V5WK0",
 "name":"Migration Schema Service",
 "displayName":"Schema Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5WK2",
 "name":"Migration Runtime Repository Data Service",
 "displayName":"Runtime Repository Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5WK3",
 "name":"Migration Application Workspace Data Service",
 "displayName":"Application Workspace Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5WK4",
 "name":"Migration Incremental Runtime Repository Data Service",
 "displayName":"Incremental Runtime Repository Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5WK5",
 "name":"Migration Incremental Application Workspace Data Service",
 "displayName":"Incremental Application Workspace Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5WK6",
 "name":"Migration Application Data Service",
 "displayName":"Application Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5WK7",
 "name":"Migration Application Data Service With Transformation",
 "displayName":"Application Data Service With Transformation",
 "type":"DB Util"
 },
 {
 "id":"88-1V5WK8",
 "name":"MigrationFilePrepareAndDeploy",
 "displayName":"File Prepare And Deploy",
 "type":"DB Util"
 },
 {
 "id":"88-1V5WK9",
 "name":"FINS BIB",
 "displayName":"FINS BIB",
 "type":"ADM"
 }
 ]
 },
 {
 "id":"88-1V5WLD",
 "name":"Prod",
 "restEndpoint":"https://slc07fnj.us.oracle.com:16690/siebel/v1.0",
 "isFavourite":"false",
 "schemaUser": "",
 "tableSpaceData": "",
 "tableSpaceIndex": "",
 "tableSpacePage16K": "",
 "tableSpacePage32K": "",
 "isUnicodeDatabase": "true",
 "resources":[
 {
 "id":"88-1V5WLE",
 "name":"Migration Schema Service",
 "displayName":"Schema Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5WLG",
 "name":"Migration Runtime Repository Data Service",
 "displayName":"Runtime Repository Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5WLH",
 "name":"Migration Application Workspace Data Service",
 "displayName":"Application Workspace Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5WLI",
 "name":"Migration Incremental Runtime Repository Data Service",
 "displayName":"Incremental Runtime Repository Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5WLJ",
 "name":"Migration Incremental Application Workspace Data Service",
 "displayName":"Incremental Application Workspace Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5WLK",
 "name":"Migration Application Data Service",
 "displayName":"Application Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5WLL",
 "name":"Migration Application Data Service With Transformation",
 "displayName":"Application Data Service With Transformation",
 "type":"DB Util"
 },
 {
 "id":"88-1V5WLM",
 "name":"MigrationFilePrepareAndDeploy",
 "displayName":"File Prepare And Deploy",
 "type":"DB Util"
 },
 {
 "id":"88-1V5WLN",
 "name":"FINS BIB",
 "displayName":"FINS BIB",
 "type":"ADM"
 }
 ]
 }
]
```

Getting a Connection by Name

You can get a connection for your migration by sending an HTTP GET request to the Siebel Migration Application.

The following details are for a request to get a connection by name for a migration:

- **URI:** `https://{hostname}:{port}/siebel/v1.0/migration/connection/{connectionName}`
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic

The following are the details for the response to a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response Body:**

```
{
 "id":"88-1V5WLD",
 "name":"Prod",
 "restEndpoint":"https://slc07fnj.us.oracle.com:16690/siebel/v1.0",
 "isFavourite":"false",
 "schemaUser": "",
 "tableSpaceData": "",
 "tableSpaceIndex": "",
 "tableSpacePage16K": "",
 "tableSpacePage32K": "",
 "isUnicodeDatabase": "true",
 "resources":[
 {
 "id":"88-1V5WLE",
 "name":"Migration Schema Service",
 "displayName":"Schema Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5WLG",
 "name":"Migration Runtime Repository Data Service",
 "displayName":"Runtime Repository Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5WLH",
 "name":"Migration Application Workspace Data Service",
 "displayName":"Application Workspace Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5WLI",
 "name":"Migration Incremental Runtime Repository Data Service",
 "displayName":"Incremental Runtime Repository Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5WLJ",
 "name":"Migration Incremental Application Workspace Data Service",
 "displayName":"Incremental Application Workspace Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5WLK",
 "name":"Migration Application Data Service",
 "displayName":"Application Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5WLL",
 "name":"Migration Application Data Service With Transformation",
 "displayName":"Application Data Service With Transformation",
 "type":"DB Util"
 },
 {
 "id":"88-1V5WLM",
 "name":"MigrationFilePrepareAndDeploy",
 "displayName":"File Prepare And Deploy",
 "type":"DB Util"
 },
 {
 "id":"88-1V5WLN",
 "name":"FINS BIB",
 "displayName":"FINS BIB",
 "type":"ADM"
 }
 ]
}
```

Creating a New Connection

You can create a new connection for your migration by sending an HTTP POST request to the Siebel Migration Application.

The following details are for a request to create a new connection for a migration:

- **URI:** `https://{hostname}:{port}/siebel/v1.0/migration/connection`
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request Body:**

```
{
 "name":"Demo Source",
 "restEndpoint":"https://{hostname}:{port}/siebel/v1.0",
 "isFavourite": "true",
 "schemaUser": "",
 "tableSpaceData": "",
 "tableSpaceIndex": "",
 "tableSpacePage16K": "",
 "tableSpacePage32K": "",
 "isUnicodeDatabase": "true"
}
```

The following are the details for the response to a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response Body:**

```
{
 "id":"88-1V5ZL2",
 "name":"Demo Source",
 "restEndpoint":"https://{hostname}:{port}/siebel/v1.0",
 "isFavourite": "true",
 "schemaUser": "",
 "tableSpaceData": "",
 "tableSpaceIndex": "",
 "tableSpacePage16K": "",
 "tableSpacePage32K": "",
 "isUnicodeDatabase": "true",
 "resources":[
 {
 "id":"88-1V5ZL3",
 "name":"Migration Schema Service",
 "displayName":"Schema Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5ZL5",
 "name":"Migration Runtime Repository Data Service",
 "displayName":"Runtime Repository Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5ZL6",
 "name":"Migration Application Workspace Data Service",
 "displayName":"Application Workspace Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5ZL7",
 "name":"Migration Incremental Runtime Repository Data Service",
 "displayName":"Incremental Runtime Repository Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5ZL8",
 "name":"Migration Incremental Application Workspace Data Service",
 "displayName":"Incremental Application Workspace Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5ZL9",
 "name":"Migration Application Data Service",
 "displayName":"Application Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5ZLA",
 "name":"Migration Application Data Service With Transformation",
 "displayName":"Application Data Service With Transformation",
 "type":"DB Util"
 },
 {
 "id":"88-1V5ZLB",
 "name":"MigrationFilePrepareAndDeploy",
 "displayName":"File Prepare And Deploy",
 "type":"DB Util"
 },
 {
 "id":"88-1V5ZLC",
 "name":"FINS BIB",
 "displayName":"FINS BIB",
 "type":"ADM"
 }
 ]
}
```

Updating the Connection

You can update a connection for your migration by sending an HTTP PUT request to the Siebel Migration Application.

The following details are for a request to update a connection for a migration:

- URI: `https://{hostname}:{port}/siebel/v1.0/migration/connection/{connectionName}`
- **HTTP Method:** PUT
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request Body:**

```
{
 "name":"Demo Dev",
 "restEndpoint":"https://{hostname}:{port}/siebel/v1.0",
 "isFavourite": "true",
 "schemaUser": "",
 "tableSpaceData": "",
 "tableSpaceIndex": "",
 "tableSpacePage16K": "",
 "tableSpacePage32K": "",
 "isUnicodeDatabase": "true"
}
```

Note: You must include all attributes and attribute values (as required) in the JSON request body. If you omit an attribute, the attribute value for that property will be updated as empty.

The following are the details for the response to a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response Body:**

```
{
 "id":"88-1V5ZL2",
 "name":"Demo Dev",
 "restEndpoint":"https://slc04ovj.us.oracle.com:5021/siebel/v1.0",
 "isFavourite": "false",
 "schemaUser": "",
 "tableSpaceData": "",
 "tableSpaceIndex": "",
 "tableSpacePage16K": "",
 "tableSpacePage32K": "",
 "isUnicodeDatabase": "true",
 "resources":[
 {
 "id":"88-1V5ZL3",
 "name":"Migration Schema Service",
 "displayName":"Schema Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5ZL5",
 "name":"Migration Runtime Repository Data Service",
 "displayName":"Runtime Repository Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5ZL6",
 "name":"Migration Application Workspace Data Service",
 "displayName":"Application Workspace Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5ZL7",
 "name":"Migration Incremental Runtime Repository Data Service",
 "displayName":"Incremental Runtime Repository Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5ZL8",
 "name":"Migration Incremental Application Workspace Data Service",
 "displayName":"Incremental Application Workspace Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5ZL9",
 "name":"Migration Application Data Service",
 "displayName":"Application Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V5ZLA",
 "name":"Migration Application Data Service With Transformation",
 "displayName":"Application Data Service With Transformation",
 "type":"DB Util"
 },
 {
 "id":"88-1V5ZLB",
 "name":"MigrationFilePrepareAndDeploy",
 "displayName":"File Prepare And Deploy",
 "type":"DB Util"
 },
 {
 "id":"88-1V5ZLC",
 "name":"FINS BIB",
 "displayName":"FINS BIB",
 "type":"ADM"
 }
 ]
}
```

Refreshing a Connection

You can refresh a connection for your migration by sending an HTTP POST request to the Siebel Migration Application.

The following details are for a request to refresh a connection for a migration:

- URI:`https://{hostname}:{port}/siebel/v1.0/migration/connection/{connectionName}/refresh`
- **HTTP Method:** POST
- **Authorization:** Basic

The following are the details for the response to a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response Body:**

```
{
 "id":"88-1V60NX",
 "name":"Demo Dev",
 "restEndpoint":"https://{hostname}:{port}/siebel/v1.0",
 "isFavourite":"true",
 "resources":[
 {
 "id":"88-1V60NY",
 "name":"Schema Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V60NZ",
 "name":"Design Repository Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V60O0",
 "name":"Runtime Repository Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V6001",
 "name":"Application Workspace Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V6002",
 "name":"Incremental Runtime Repository Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V6003",
 "name":"Incremental Application Workspace Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V6004",
 "name":"Application Data Service",
 "type":"DB Util"
 },
 {
 "id":"88-1V6005",
 "name":"Application Data Service With Transformation",
 "type":"DB Util"
 },
 {
 "id":"88-1V6006",
 "name":"File Prepare And Deploy",
 "type":"DB Util"
 },
 {
 "id":"88-1V6007",
 "name":"FINS BIB",
 "type":"ADM"
 },
 {
 "id":"88-1V6008",
 "name":"ADM AG",
 "type":"ADM"
 },
 {
 "id":"88-1V6009",
 "name":"AccGrp",
 "type":"ADM"
 }
 ]
}
```

Creating a Watermark for a Connection

You can create a watermark for your migration connection by sending an HTTP POST request to the Siebel Migration Application.

The following details are for a request to create a watermark for a migration connection:

- **URI:** `https://{hostname}:{port}/siebel/v1.0/migration/connection/{connectionName}/watermark`
- **HTTP Method**: POST
- **Content-Type:** application/json
- **Authorization:** Basic

- **Request Body:**

```
{
 "fileName":"demo.txt"
}
```

The following are the details for the response to a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response Body:**

```
{
 "watermarkFile":"c:\\fs\\migration\\demo.txt"
}
```

Deleting a Connection

You can delete a migration connection by sending an HTTP DELETE request to the Siebel Migration Application.

The following details are for a request to delete a migration connection:

- **URI:** `https://{hostname}:{port}/siebel/v1.0/migration/connection/{connectionName}`
- **HTTP Method:** DELETE
- **Content-Type:** application/json
- **Authorization:** Basic

Using REST API to Configure Siebel Migration Application Migration Plans

You can use the Siebel Migration REST API to configure your migration plan. For more information, see the following subtopics:

- *Getting All Migration Plans*
- *Getting a Migration Plan by Name*
- *Creating a New Migration Plan*
- *Updating a Migration Plan*
- *Refreshing a Migration Plan*
- *Deleting a Migration Plan*

Getting All Migration Plans

You can get all migration plans for your migration by sending an HTTP GET request to the Siebel Migration Application.

The following details are for a request to get all migration plans for a migration:

- **URI:** `https://{hostname}:{port}/siebel/v1.0/migration/plan`
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic

The following are the details for the response to a successful request:

- **HTTP Code**: 200
- **Content-Type:** application/json
- **Response Body:**

```
[
 {
 "id":"88-1V5WMR",
 "name":"IRRMigration",
 "description":"IRRMigration",
 "source":"Dev",
 "target":"Prod",
 "resources":[
 {
 "id":"88-1V5WMS",
 "name":"Migration Schema Service",
 "displayName":"Schema Service",
 "isSelected":"true",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V5WMU",
 "name":"Migration Runtime Repository Data Service",
 "displayName":"Runtime Repository Data Service",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V5WMV",
 "name":"Migration Application Workspace Data Service",
 "displayName":"Application Workspace Data Service",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V5WMW",
 "name":"Migration Incremental Runtime Repository Data Service",
 "displayName":"Incremental Runtime Repository Data Service",
 "isSelected":"true",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V5WMX",
 "name":"Migration Incremental Application Workspace Data Service",
 "displayName":"Incremental Application Workspace Data Service",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
```

```
"id":"88-1V5WMY",
"name":"Migration Application Data Service",
"displayName":"Application Data Service",
"isSelected":"false",
"integrationBranchName":"",
"versionNumber":"0",
"language":"",
"sequenceNumber":"0"
},
{
"id":"88-1V5WMZ",
"name":"Migration Application Data Service With Transformation",
"displayName":"Application Data Service With Transformation",
"isSelected":"false",
"integrationBranchName":"",
"versionNumber":"0",
"language":"",
"sequenceNumber":"0"
},
{
"id":"88-1V5WN0",
"name":"MigrationFilePrepareAndDeploy",
"displayName":"File Prepare And Deploy",
"isSelected":"false",
"integrationBranchName":"",
"versionNumber":"0",
"language":"",
"sequenceNumber":"0"
},
{
"id":"88-1V5WN1",
"name":"FINS BIB",
"displayName":"FINS BIB",
"isSelected":"false",
"integrationBranchName":"",
"versionNumber":"0",
"language":"",
"sequenceNumber":"0"
}
]
},
{
"id":"88-1V5Y2U",
"name":"Data Mig",
"description":"Data Mig",
"source":"Dev",
"target":"Prod",
"resources":[
{
"id":"88-1V5Y2V",
"name":"Migration Schema Service",
"displayName":"Schema Service",
"isSelected":"false",
"integrationBranchName":"",
"versionNumber":"0",
"language":"",
"sequenceNumber":"0"
},
{
"id":"88-1V5Y2X",
"name":"Migration Runtime Repository Data Service",
"displayName":"Runtime Repository Data Service",
"isSelected":"false",
"integrationBranchName":"",
"versionNumber":"0",
"language":"",
"sequenceNumber":"0"
},
{
"id":"88-1V5Y2Y",
"name":"Migration Application Workspace Data Service",
"displayName":"Application Workspace Data Service",
"isSelected":"false",
"integrationBranchName":"",
"versionNumber":"0",
"language":"",
"sequenceNumber":"0"
},
{
"id":"88-1V5Y2Z",
"name":"Migration Incremental Runtime Repository Data Service",
"displayName":"Incremental Runtime Repository Data Service",
"isSelected":"false",
"integrationBranchName":"",
"versionNumber":"0",
"language":"",
"sequenceNumber":"0"
},
{
"id":"88-1V5Y30",
"name":"Migration Incremental Application Workspace Data Service",
"displayName":"Incremental Application Workspace Data Service",
"isSelected":"false",
"integrationBranchName":"",
"versionNumber":"0",
"language":"",
"sequenceNumber":"0"
},
{
"id":"88-1V5Y31",
"name":"Migration Application Data Service",
"displayName":"Application Data Service",
"isSelected":"true",
"integrationBranchName":"",
"versionNumber":"0",
"language":"",
"sequenceNumber":"0"
},
{
"id":"88-1V5Y32",
"name":"Migration Application Data Service With Transformation",
"displayName":"Application Data Service With Transformation",
"isSelected":"false",
"integrationBranchName":"",
"versionNumber":"0",
"language":"",
"sequenceNumber":"0"
},
{
"id":"88-1V5Y33",
"name":"MigrationFilePrepareAndDeploy",
"displayName":"File Prepare And Deploy",
"isSelected":"false",
"integrationBranchName":"",
"versionNumber":"0",
"language":"",
"sequenceNumber":"0"
},
{
"id":"88-1V5Y34",
"name":"FINS BIB",
"displayName":"FINS BIB",
"isSelected":"false",
"integrationBranchName":"",
"versionNumber":"0",
"language":"",
"sequenceNumber":"0"
}
]
}
]
```

Getting a Migration Plan by Name

You can get a migration plan by name for your migration by sending an HTTP GET request to the Siebel Migration Application.

The following details are for a request to get a migration plan by name:

- **URI:** `https://{hostname}:{port}/siebel/v1.0/migration/plan/{planName}`
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic

The following are the details for the response to a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response Body:**

```
{
"id":"88-1V5WMR",
"name":"IRRMigration",
"description":"IRRMigration",
"source":"Dev",
"target":"Prod",
"resources":[
{
"id":"88-1V5WMS",
"name":"Migration Schema Service",
"displayName":"Schema Service",
"isSelected":"true",
"integrationBranchName":"",
"versionNumber":"0",
"language":"",
"sequenceNumber":"0"
},
{
"id":"88-1V5WMU",
"name":"Migration Runtime Repository Data Service",
"displayName":"Runtime Repository Data Service",
"isSelected":"false",
"integrationBranchName":"",
"versionNumber":"0",
"language":"",
"sequenceNumber":"0"
},
{
"id":"88-1V5WMV",
"name":"Migration Application Workspace Data Service",
"displayName":"Application Workspace Data Service",
"isSelected":"false",
"integrationBranchName":"",
"versionNumber":"0",
"language":"",
"sequenceNumber":"0"
},
{
"id":"88-1V5WMW",
"name":"Migration Incremental Runtime Repository Data Service",
"displayName":"Incremental Runtime Repository Data Service",
"isSelected":"true",
"integrationBranchName":"",
"versionNumber":"0",
"language":"",
"sequenceNumber":"0"
},
{
"id":"88-1V5WMX",
"name":"Migration Incremental Application Workspace Data Service",
"displayName":"Incremental Application Workspace Data Service",
"isSelected":"false",
"integrationBranchName":"",
"versionNumber":"0",
"language":"",
"sequenceNumber":"0"
},
{
"id":"88-1V5WMY",
"name":"Migration Application Data Service",
"displayName":"Application Data Service",
"isSelected":"false",
"integrationBranchName":"",
"versionNumber":"0",
"language":"",
"sequenceNumber":"0"
},
{
"id":"88-1V5WMZ",
"name":"Migration Application Data Service With Transformation",
"displayName":"Application Data Service With Transformation",
"isSelected":"false",
"integrationBranchName":"",
"versionNumber":"0",
"language":"",
"sequenceNumber":"0"
},
{
"id":"88-1V5WN0",
"name":"MigrationFilePrepareAndDeploy",
"displayName":"File Prepare And Deploy",
"isSelected":"false",
"integrationBranchName":"",
"versionNumber":"0",
"language":"",
"sequenceNumber":"0"
},
{
"id":"88-1V5WN1",
"name":"FINS BIB",
"displayName":"FINS BIB",
"isSelected":"false",
"integrationBranchName":"",
"versionNumber":"0",
"language":"",
"sequenceNumber":"0"
}
]
},
```

Creating a New Migration Plan

You can create a new migration plan for your migration by sending an HTTP POST request to the Siebel Migration Application.

The following details are for a request to create a new migration plan for a migration:

- **URI:** `https://{hostname}:{port}/siebel/v1.0/migration/plan`
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request Body:**

```
{
 "name":"Demo Data Mig",
 "description":"Demo Data Migration",
 "source":"Dev",
 "target":"Prod",
 "resources":[
 {
 "id":"88-1V5WLP",
 "name":"Schema Service",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V5WLR",
 "name":"Runtime Repository Data Service",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V5WLS",
 "name":"Application Workspace Data Service",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V5WLT",
 "name":"Incremental Runtime Repository Data Service",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V5WLU",
 "name":"Incremental Application Workspace Data Service",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V5WLV",
 "name":"Application Data Service",
 "isSelected":"true",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V5WLW",
 "name":"Application Data Service With Transformation",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V5WLX",
 "name":"File Prepare And Deploy",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V5WLY",
 "name":"FINS BIB",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 }
 ]
}
```

The following are the details for the response to a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response Body:**

```
{
 "name":"Demo Data Mig",
 "description":"Demo Data Migration",
 "source":"Dev",
 "target":"Prod",
 "resources":[
 {
 "id":"88-1V5WLP",
 "name":"Schema Service",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V5WLR",
 "name":"Runtime Repository Data Service",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V5WLS",
 "name":"Application Workspace Data Service",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V5WLT",
 "name":"Incremental Runtime Repository Data Service",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V5WLU",
 "name":"Incremental Application Workspace Data Service",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V5WLV",
 "name":"Application Data Service",
 "isSelected":"true",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V5WLW",
 "name":"Application Data Service With Transformation",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V5WLX",
 "name":"File Prepare And Deploy",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V5WLY",
 "name":"FINS BIB",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 }
 ]
}
```

Updating a Migration Plan

You can update a migration plan for your migration by sending an HTTP PUT request to the Siebel Migration Application.

The following details are for a request to update a migration plan for a migration:

- **URI:** `https://{hostname}:{port}/siebel/v1.0/migration/plan/{planName}`
- **HTTP Method:** PUT
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request Body:**

```
{
  "name":"Demo Data Migration Plan",
  "description":"Demo Data Migration for Winter",
  "source":"Dev",
  "target":"Prod",
  "resources":[
  {
  "name":"Migration Schema Service",
  "displayName":"Schema Service",
  "isSelected":"false",
  "integrationBranchName":"",
  "versionNumber":"0",
  "language":"",
  "sequenceNumber":"0"
  },
  {
  "name":"Migration Runtime Repository Data Service",
  "displayName":"Runtime Repository Data Service",
  "isSelected":"false",
  "integrationBranchName":"",
  "versionNumber":"0",
  "language":"",
  "sequenceNumber":"0"
  },
  {
  "name":"Migration Application Workspace Data Service",
  "displayName":"Application Workspace Data Service",
  "isSelected":"false",
  "integrationBranchName":"",
  "versionNumber":"0",
  "language":"",
  "sequenceNumber":"0"
  },
  {
  "name":"Migration Incremental Runtime Repository Data Service",
  "displayName":"Incremental Runtime Repository Data Service",
  "isSelected":"false",
  "integrationBranchName":"",
  "versionNumber":"0",
  "language":"",
  "sequenceNumber":"0"
  },
  {
  "name":"Migration Incremental Application Workspace Data Service",
  "displayName":"Incremental Application Workspace Data Service",
  "isSelected":"false",
  "integrationBranchName":"",
  "versionNumber":"0",
  "language":"",
  "sequenceNumber":"0"
  },
  {
  "name":"Migration Application Data Service",
  "displayName":"Application Data Service",
  "isSelected":"true",
  "integrationBranchName":"",
  "versionNumber":"0",
  "language":"",
  "sequenceNumber":"0"
  },
  {
  "name":"Migration Application Data Service With Transformation",
  "displayName":"Application Data Service With Transformation",
  "isSelected":"false",
  "integrationBranchName":"",
  "versionNumber":"0",
  "language":"",
  "sequenceNumber":"0"
  },
  {
  "name":"MigrationFilePrepareAndDeploy",
  "displayName":"File Prepare And Deploy",
  "isSelected":"true",
  "integrationBranchName":"",
  "versionNumber":"0",
  "language":"",
  "sequenceNumber":"0"
  },
  {
  "name":"FINS BIB",
  "displayName":"FINS BIB",
  "isSelected":"false",
  "integrationBranchName":"",
  "versionNumber":"0",
  "language":"",
  "sequenceNumber":"0"
  }
  ]
 }
```

The following are the details for the response to a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response Body:** None

Refreshing a Migration Plan

You can refresh a migration plan for your migration by sending an HTTP POST request to the Siebel Migration Application.

The following details are for a request to refresh a migration plan:

- URI:`https://{hostname}:{port}/siebel/v1.0/migration/plan/{planName}/refresh`
- **HTTP Method:** POST
- **Authorization:** Basic

The following are the details for the response to a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response Body:**

```
{
 "id":"88-1V6008",
 "name":"Data Migration for Winter",
 "description":"Data Migration for Winter Release Branch",
 "source":"Dev",
 "target":"Prod",
 "resources":[
 {
 "id":"88-1V6009",
 "name":"Schema Service",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V600A",
 "name":"Design Repository Data Service",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V600B",
 "name":"Runtime Repository Data Service",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V600C",
 "name":"Application Workspace Data Service",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V600D",
 "name":"Incremental Runtime Repository Data Service",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V600E",
 "name":"Incremental Application Workspace Data Service",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V60OF",
 "name":"Application Data Service",
 "isSelected":"true",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V60OG",
 "name":"Application Data Service With Transformation",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V60OH",
 "name":"File Prepare And Deploy",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V60OI",
 "name":"FINS BIB",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V60OJ",
 "name":"ADM AG",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 },
 {
 "id":"88-1V60OK",
 "name":"Access Group",
 "isSelected":"false",
 "integrationBranchName":"",
 "versionNumber":"0",
 "language":"",
 "sequenceNumber":"0"
 }
 ]
}
```

Deleting a Migration Plan

You can delete for your migration by sending an HTTP DELETE request to the Siebel Migration Application.

The following details are for a request to delete a migration plan for a migration:

- **URI:** `https://{hostname}:{port}/siebel/v1.0/migration/plan/{planName}`
- **HTTP Method:** DELETE
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request Body:**

  None

The following are the details for the response to a successful request:

- **HTTP Code:** 200
- **Response Body:** None

Using REST API to Execute Schema Imports

You can manually run schema imports using the Migration Schema Service Import Business Service from inbound REST requests. This is one of the steps that happen automatically during a migration invoked using the Migration Application, but you can run the steps and services individually.

When running the service, care must be taken when providing the Schema Owner Password used in the request.

We allow the Schema Owner Password to be encrypted using the encryptstring utility or passed as a Base64 encoded string but not both. Encrypting the password using the encryptstring utility is optional but recommended. When you do wish to use an encrypted password for inbound REST calls to execute the MigrationSchemaServiceImport Business Service, you use the following parameter:

pwdPreEncrypted

Examples:

```
“pwdPreEncrypted”:”true” (password has been encrypted)
“pwdPreEncrypted”:”false” (password has not been encrypted)
```

Setting this parameter to **true** tells the Siebel Migration Application that you have encrypted the Schema Owner Password using the encryptstring utility and are passing that encrypted password in the inbound REST call. Setting it to **false** means you are passing a Base64 encoded password.

In order to make sure you can successfully use either strategy, follow the steps listed below.

Steps

If you wish to run the MigrationSchemaServiceImport Business Service using a Base64 encoded password set the pwdPreEncrypted parameter to **false** and provide the Base64 encoded password.

In this example the password is Base64 encoded and the pwdPreEncrypted parameter is set to **false**.

```
"migrationid":"88-38ZVN4",
"password":"b3JhNjYzMTEy",
"pwdPreEncrypted":"false",
"filename":"ZGRsZXhwb3J0XzNBMERGOTdCNDdGQS5sb2c=",
"isUnicodeDatabase":"Y",
"username":"ora663112"}}
```

If you wish to run the MigrationSchemaServiceImport Business Service using an encrypted password set the pwdPreEncrypted parameter to **true** and provide the encrypted password.

In this example the password has already been encrypted and the pwdPreEncrypted parameter is set to **true**.

```
{"body":{
"migrationid":"88-38ZVR5",
"password":"JvJbl4ZukuAV0JRnLpneNQXv3gAA",
"pwdPreEncrypted":"true",
"filename":"ZGRsZXhwb3J0XzNBMERGQkVDRDNGNi5sb2c",
"isUnicodeDatabase":"Y",
"username":"ora663112"}}
```

Note: Your encrypted password is passed as is. Don't further encode it (for example don't take an encrypted password and then Base64 encode it) or you'll receive an error.

Using REST API to Execute Siebel Migration Plans

You can use the Siebel Migration REST API to execute migration plans. For more information, see the following subtopics:

- *Executing a Migration Plan*
- *Getting Status for a Running Migration Plan by Name*
- *Getting Status for a Running Migration Plan by Plan Name and Resource Name*
- *Getting Execution Status by Migration Plan Name, Resource Name, and Operation*
- *Getting the Migration Execution Operation Log*

Executing a Migration Plan

You can execute a migration plan for your migration by sending an HTTP POST request to the Siebel Migration Application.

Depending on the migration service that you selected in your migration plan, you must provide the following additional information in your REST API request:

- If you selected a Schema Service with Export & Import or Import only for your migration plan, then you must provide the following parameter and value in your REST API request:
    - **schemaPassword.** The password for the schema user. The schemaPassword must be encrypted by using base64 in the REST API request if pre-Encrypted is not used.

      Note: If you want to use a pre-encrypted Schema Owner Password encrypted using the encryptstring.exe utility, don't use the base64 encoded password. Use your pre-encrypted password, add the parameter *pwdPreEncrypted* and set its value to **true**. This tells the migration process that your password has already been encrypted using encryptstring and you don't need to encrypt it again during the process. Leaving the parameter out or setting it to false indicates that you are using a base64 encoded password for the REST call.

- If you selected the Incremental Runtime Repository or the Incremental Application Workspace Data Service or the File Prepare and Deploy Service with Export only for your migration plan, then you must provide the watermarkFilename parameter and value in your REST API request.
- If you want to only migrate a specific version when run the Incremental Runtime Repository (Export & Import or Export only) migration, then you must provide the irrEndVersion parameter and value in the REST API request. If you do not provide the irrEndVersion parameter and value, then, the migration migrates up to the latest version.
- For all Export only or Import only migration plans, you must provide the packageFilename parameter and value in the REST API request.

The following details are for a request to execute a migration plan:

- **URI:** `https://{hostname}:{port}/siebel/v1.0/migration/execution/{planName}`
- **HTTP Method:** POST
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request Body:**

```
{
"schemaPassword":"RA8c4/CVDs9KOsrHmzW+xwX4kXwi",
"pwdPreEncrypted":"true",
"watermarkFilename":"",
"irrEndVersion":"",
"packageFilename":"",
}
```

Note: If schemaUser or isUnicodeDatabase (which are part of the Input payload) are provided in the Request Body, then they will be ignored. Instead, schemaUser and isUnicodeDatabase will be read from the Target connection. If neither schemaUser nor isUnicodeDatabase are specified in the Target connection, then an error will be returned.

The following are the details for the response to a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response Body:**

```
{
 "id":"88-1V5YFC",
 "planName":"Data Mig",
 "description":"Data Mig",
 "status":"Running",
 "startDate":"2018-07-24 09:50:20",
 "endDate":"",
 "source":"Dev",
 "target":"Prod",
 "packageName":"",
 "resources":[
 {
 "id":"88-1V5YFD",
 "name":"",
 "operation":"Export",
 "sequenceNumber":"1",
 "mode":"Asynchronous",
 "status":"Not Started",
 "startTime":"",
 "endTime":"",
 "resourceType":""
 },
 {
 "id":"88-1V5YFE",
 "name":"",
 "operation":"Import",
 "sequenceNumber":"2",
 "mode":"Asynchronous",
 "status":"Not Started",
 "startTime":"",
 "endTime":"",
 "resourceType":""
 }
 ]
}
```

Getting Status for a Running Migration Plan by Name

You can get status for a running migration plan by name for your migration by sending an HTTP GET request to the Siebel Migration Application.

The following details are for a request to get status for a migration plan by name for a migration:

- **URI:** `https://{hostname}:{port}/siebel/v1.0/migration/execution/{planName}`
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request Body:** None

The following are the details for the response to a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response Body:**

```
{
 "id":"88-1V5WPF",
 "planName":"IRRMigration",
 "description":"IRR Migration",
 "status":"Running",
 "startDate":"2018-07-17 04:33:54",
 "endDate":"",
 "source":"Dev",
 "target":"Prod",
 "packageName":"",
 "resources":[
 {
 "id":"88-1V5WPG",
 "name":"",
 "operation":"Export",
 "sequenceNumber":"1",
 "mode":"Asynchronous",
 "status":"Not Applicable",
 "startTime":"",
 "endTime":"",
 "resourceType":""
 },
 {
 "id":"88-1V5WPH",
 "name":"",
 "operation":"Import",
 "sequenceNumber":"2",
 "mode":"Asynchronous",
 "status":"Not Applicable",
 "startTime":"",
 "endTime":"",
 "resourceType":""
 },
 {
 "id":"88-1V5WPI",
 "name":"",
 "operation":"Export",
 "sequenceNumber":"3",
 "mode":"Asynchronous",
 "status":"Running",
 "startTime":"2018-07-17 04:33:55",
 "endTime":"",
 "resourceType":""
 },
 {
 "id":"88-1V5WPJ",
 "name":"",
 "operation":"Import",
 "sequenceNumber":"4",
 "mode":"Asynchronous",
 "status":"Not Started",
 "startTime":"",
 "endTime":"",
 "resourceType":""
 },
 {
 "id":"88-1V5WPK",
 "name":"",
 "sequenceNumber":"5",
 "mode":"Asynchronous",
 "status":"Not Started",
 "startTime":"",
 "endTime":"",
 "resourceType":""
 },
 {
 "id":"88-1V5WPL",
 "name":"",
 "operation":"DBCheck",
 "sequenceNumber":"6",
 "mode":"Asynchronous",
 "status":"Not Started",
 "startTime":"",
 "endTime":"",
 "resourceType":""
 }
 ]
}
```

Getting Status for a Running Migration Plan by Plan Name and Resource Name

You can get status for a running migration plan by plan name and resource name for your migration by sending an HTTP GET request to the Siebel Migration Application.

The following details are for a request to get status for a running migration plan by plan name and resource name for a migration:

- **URI:** `https://{hostname}:{port}/siebel/v1.0/migration/execution/{planName}/{resourceName}`
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request Body:** None

The following are the details for the response to a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response Body:**

```
{
 "id":"88-1V60OJ",
 "planName":"Repo Export",
 "description":"Repo Export",
 "status":"Running",
 "startDate":"2018-08-17 06:02:39",
 "endDate":"",
 "source":"Dev",
 "target":"",
 "packageName":"irr_export.zip",
 "resources":[
 {
 "id":"88-1V60OK",
 "name":"Schema Service",
 "operation":"Export",
 "sequenceNumber":"1",
 "mode":"Asynchronous",
 "status":"Not Applicable",
 "startTime":"",
 "endTime":"",
 "resourceType":"DB Util"
 },
 {
 "id":"88-1V61H2",
 "name":"Incremental Runtime Repository Data Service",
 "operation":"Export",
 "sequenceNumber":"2",
 "mode":"Asynchronous",
 "status":"Running",
 "startTime":"2018-08-17 06:02:39",
 "endTime":"",
 "resourceType":"DB Util"
 }
 ]
}
```

Getting Execution Status by Migration Plan Name, Resource Name and Operation

You can get execution status by migration plan name, resource name, and operation for your migration by sending an HTTP GET request to the Siebel Migration Application.

The following details are for a request to get execution status by migration plan name, resource name, and operation for a migration:

- **URI:** `https://{hostname}:{port}/siebel/v1.0/migration/execution/{planName}/{resourceName}/{operation}`
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request Body:** None

The following are the details for the response to a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response Body:**

```
{
 "id":"88-1V5WPF",
 "planName":"Test REST Execution",
 "description":"Test REST Execution",
 "status":"Running",
 "startDate":"2018-07-17 04:33:54",
 "endDate":"",
 "source":"Dev",
 "target":"Prod",
 "packageName":"",
 "resources":[
 {
 "id":"88-1V5WPI",
 "name":"Incremental Runtime Repository Data Service",
 "operation":"Export",
 "sequenceNumber":"3",
 "mode":"Asynchronous",
 "status":"Running",
 "startTime":"2018-07-17 04:33:55",
 "endTime":"",
 "resourceType":""
 }
 ]
}
```

Getting the Migration Execution Operation Log

You can get the migration execution operation log for your migration by sending an HTTP GET request to the Siebel Migration Application.

The following details are for a request to get the migration execution operation log for a migration:

- **URI:** `https://{hostname}:{port}/siebel/v1.0/migration/execution/{planName}/{resourceName}/{operation}/log`
- **HTTP Method:** GET
- **Content-Type:** application/json

- **Authorization:** Basic
- **Request Body:** None

The response returns the migration execution operation log for your migration. The following are the details for the response to a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response Body:**

Using REST API to Get Siebel Migration Plan History

You can use the Siebel Migration REST API to get migration plan history. For more information, see the following subtopics:

- *Getting All Migration History*
- *Getting Migration History by Migration Plan Name*
- *Getting Migration History by ID Number*
- *Getting Migration History by ID and Resource Name*
- *Getting Migration History by ID, Resource Name, and Operation*
- *Getting the Migration History Operation Log*

Getting All Migration History

You can get all the history for your migration by sending an HTTP GET request to the Siebel Migration Application.

The following details are for a request to get all the history for a migration:

- **URI:** `https://{hostname}:{port}/siebel/v1.0/migration/history`
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request Body:** None

The following are the details for the response to a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response Body:**

```
[
 {
 "id":"88-1V5YFC",
 "planName":"Data Mig",
 "description":"Data Mig",
 "status":"Success",
 "startDate":"2018-07-24 09:50:20",
 "endDate":"2018-07-24 09:51:07",
 "source":"Dev",
 "target":"Prod",
 "packageName":"",
 "resources":[
 {
```

```
"id":"88-1V5YFD",
"name":"Application Data Service",
"operation":"Export",
"sequenceNumber":"1",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-24 09:50:20",
"endTime":"2018-07-24 09:50:36",
"resourceType":"DB Util"
},
{
"id":"88-1V5YFE",
"name":"Application Data Service",
"operation":"Import",
"sequenceNumber":"2",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-24 09:50:52",
"endTime":"2018-07-24 09:51:07",
"resourceType":"DB Util"
}
]
},
{
"id":"88-1V5Y4X",
"planName":"Data Mig",
"description":"Data Mig",
"status":"Success",
"startDate":"2018-07-23 02:47:14",
"endDate":"2018-07-23 02:48:00",
"source":"Dev",
"target":"Prod",
"packageName":"",
"resources":[
{
"id":"88-1V5Y4Y",
"name":"Application Data Service",
"operation":"Export",
"sequenceNumber":"1",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-23 02:47:14",
"endTime":"2018-07-23 02:47:29",
"resourceType":"DB Util"
},
{
"id":"88-1V5Y4Z",
"name":"Application Data Service",
"operation":"Import",
"sequenceNumber":"2",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-23 02:47:44",
"endTime":"2018-07-23 02:47:59",
"resourceType":"DB Util"
}
]
},
{
"id":"88-1V5WOX",
"planName":"IRRMigration",
"description":"IRRMigration",
"status":"Error",
"startDate":"2018-07-17 04:29:06",
"endDate":"2018-07-17 04:29:22",
"source":"Dev",
"target":"Prod1",
"packageName":"",
"resources":[
{
"id":"88-1V5WOY",
"name":"Schema Service",
"operation":"Export",
"sequenceNumber":"1",
"mode":"Asynchronous",
"status":"Not Applicable",
"startTime":"",
"endTime":"",
"resourceType":"DB Util"
},
{
"id":"88-1V5WOZ",
"name":"Schema Service",
"operation":"Import",
"sequenceNumber":"2",
"mode":"Asynchronous",
"status":"Not Applicable",
"startTime":"",
"endTime":"",
"resourceType":"DB Util"
},
{
"id":"88-1V5WP0",
"name":"Incremental Runtime Repository Data Service",
"operation":"Export",
"sequenceNumber":"3",
"mode":"Asynchronous",
"status":"Error",
"startTime":"2018-07-17 04:29:07",
"endTime":"2018-07-17 04:29:22",
"resourceType":"DB Util"
},
{
"id":"88-1V5WP1",
"name":"Incremental Runtime Repository Data Service",
"operation":"Import",
"sequenceNumber":"4",
"mode":"Asynchronous",
"status":"Not Started",
"startTime":"",
"endTime":"",
"resourceType":"DB Util"
},
{
"id":"88-1V5WP2",
"name":"Incremental Runtime Repository Data Service",
"sequenceNumber":"5",
"mode":"Asynchronous",
"status":"Not Started",
"startTime":"",
"endTime":"",
"resourceType":"DB Util"
},
{
"id":"88-1V5WP3",
"name":"Incremental Runtime Repository Data Service",
"operation":"DBCheck",
"sequenceNumber":"6",
"mode":"Asynchronous",
"status":"Not Started",
"startTime":"",
"endTime":"",
"resourceType":"DB Util"
}
]
},
{
"id":"88-1V5WOU",
"planName":"Rest Testing",
"description":"Rest Testing",
"status":"Error",
"startDate":"2018-07-17 04:28:35",
"endDate":"2018-07-17 04:28:51",
"source":"Dev",
"target":"Prod1",
"packageName":"",
"resources":[
{
"id":"88-1V5WOV",
"name":"Application Data Service",
"operation":"Export",
"sequenceNumber":"1",
"mode":"Asynchronous",
"status":"Error",
"startTime":"2018-07-17 04:28:35",
"endTime":"2018-07-17 04:28:51",
"resourceType":"DB Util"
},
{
"id":"88-1V5WOW",
"name":"Application Data Service",
"operation":"Import",
"sequenceNumber":"2",
"mode":"Asynchronous",
"status":"Not Started",
"startTime":"",
"endTime":"",
"resourceType":"DB Util"
}
]
},
{
"id":"88-1V5WON",
"planName":"IRRMigration",
"description":"IRRMigration",
"status":"Error",
"startDate":"2018-07-17 04:28:06",
"endDate":"2018-07-17 04:28:22",
"source":"Dev",
"target":"Prod1",
"packageName":"",
"resources":[
{
"id":"88-1V5WOO",
"name":"Schema Service",
"operation":"Export",
"sequenceNumber":"1",
"mode":"Asynchronous",
"status":"Not Applicable",
"startTime":"",
"endTime":"",
"resourceType":"DB Util"
},
{
"id":"88-1V5WOP",
"name":"Schema Service",
"operation":"Import",
"sequenceNumber":"2",
        "mode":"Asynchronous",
        "status":"Not Applicable",
        "startTime":"",
        "endTime":"",
        "resourceType":"DB Util"
        },
        {
        "id":"88-1V5WOQ",
        "name":"Incremental Runtime Repository Data Service",
        "operation":"Export",
        "sequenceNumber":"3",
        "mode":"Asynchronous",
        "status":"Error",
        "startTime":"2018-07-17 04:28:07",
        "endTime":"2018-07-17 04:28:22",
        "resourceType":"DB Util"
        },
        {
        "id":"88-1V5WOR",
        "name":"Incremental Runtime Repository Data Service",
        "operation":"Import",
        "sequenceNumber":"4",
        "mode":"Asynchronous",
        "status":"Not Started",
        "startTime":"",
        "endTime":"",
        "resourceType":"DB Util"
        },
        {
        "id":"88-1V5WOS",
        "name":"Incremental Runtime Repository Data Service",
        "sequenceNumber":"5",
        "mode":"Asynchronous",
        "status":"Not Started",
        "startTime":"",
        "endTime":"",
        "resourceType":"DB Util"
        },
        {
        "id":"88-1V5WOT",
        "name":"Incremental Runtime Repository Data Service",
        "operation":"DBCheck",
        "sequenceNumber":"6",
        "mode":"Asynchronous",
        "status":"Not Started",
        "startTime":"",
        "endTime":"",
        "resourceType":"DB Util"
        }
        ]
        }
       ]
```

Getting Migration History by Migration Plan Name

You can get the migration history by migration plan name for your migration by sending an HTTP GET request to the Siebel Migration Application.

The following details are for a request to get the migration history by migration plan name for a migration:

- **URI:** `https://{hostname}:{port}/siebel/v1.0/migration/history?plan=<planName>`
- **HTTP Method:** GET
- **Content-Type:** application/json

- **Authorization:** Basic
- **Request Body:** None

The following are the details for the response to a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response Body:**

```
[
 {
 "id":"88-1V5YFF",
 "planName":"Data Mig",
 "description":"Data Mig",
 "status":"Success",
 "startDate":"2018-07-24 08:44:42",
 "endDate":"2018-07-24 08:45:27",
 "source":"Dev",
 "target":"Prod",
 "packageName":"",
 "resources":[
 {
 "id":"88-1V5YFG",
 "name":"Application Data Service",
 "operation":"Export",
 "sequenceNumber":"1",
 "mode":"Asynchronous",
 "status":"Success",
 "startTime":"2018-07-24 08:44:42",
 "endTime":"2018-07-24 08:44:57",
 "resourceType":"DB Util"
 },
 {
 "id":"88-1V5YFH",
 "name":"Application Data Service",
 "operation":"Import",
 "sequenceNumber":"2",
 "mode":"Asynchronous",
 "status":"Success",
 "startTime":"2018-07-24 08:45:12",
 "endTime":"2018-07-24 08:45:27",
 "resourceType":"DB Util"
 }
 ]
 },
 {
 "id":"88-1V5YFC",
 "planName":"Data Mig",
 "description":"Data Mig",
 "status":"Success",
 "startDate":"2018-07-24 09:50:20",
 "endDate":"2018-07-24 09:51:07",
 "source":"Dev",
 "target":"Prod",
 "packageName":null,
 "resources":[
 {
 "id":"88-1V5YFD",
 "name":"Application Data Service",
 "operation":"Export",
 "sequenceNumber":"1",
 "mode":"Asynchronous",
 "status":"Success",
 "startTime":"2018-07-24 09:50:20",
 "endTime":"2018-07-24 09:50:36",
"resourceType":"DB Util"
},
{
"id":"88-1V5YFE",
"name":"Application Data Service",
"operation":"Import",
"sequenceNumber":"2",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-24 09:50:52",
"endTime":"2018-07-24 09:51:07",
"resourceType":"DB Util"
}
]
},
{
"id":"88-1V5YB6",
"planName":"Data Mig",
"description":"Data Mig",
"status":"Success",
"startDate":"2018-07-23 09:37:39",
"endDate":"2018-07-23 09:38:24",
"source":"Dev",
"target":"Prod",
"packageName":null,
"resources":[
{
"id":"88-1V5YB7",
"name":"Application Data Service",
"operation":"Export",
"sequenceNumber":"1",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-23 09:37:39",
"endTime":"2018-07-23 09:37:54",
"resourceType":"DB Util"
},
{
"id":"88-1V5YB8",
"name":"Application Data Service",
"operation":"Import",
"sequenceNumber":"2",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-23 09:38:09",
"endTime":"2018-07-23 09:38:24",
"resourceType":"DB Util"
}
]
},
{
"id":"88-1V5YAK",
"planName":"Data Mig",
"description":"Data Mig",
"status":"Success",
"startDate":"2018-07-23 09:09:48",
"endDate":"2018-07-23 09:10:34",
"source":"Dev",
"target":"Prod",
"packageName":"",
"resources":[
{
"id":"88-1V5YAL",
"name":"Application Data Service",
"operation":"Export",
"sequenceNumber":"1",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-23 09:09:48",
"endTime":"2018-07-23 09:10:03",
"resourceType":"DB Util"
},
{
"id":"88-1V5YAM",
"name":"Application Data Service",
"operation":"Import",
"sequenceNumber":"2",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-23 09:10:19",
"endTime":"2018-07-23 09:10:34",
"resourceType":"DB Util"
}
]
},
{
"id":"88-1V5YAH",
"planName":"Data Mig",
"description":"Data Mig",
"status":"Success",
"startDate":"2018-07-23 07:19:46",
"endDate":"2018-07-23 07:20:32",
"source":"Dev",
"target":"Prod",
"packageName":null,
"resources":[
{
"id":"88-1V5YAI",
"name":"Application Data Service",
"operation":"Export",
"sequenceNumber":"1",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-23 07:19:46",
"endTime":"2018-07-23 07:20:01",
"resourceType":"DB Util"
},
{
"id":"88-1V5YAJ",
"name":"Application Data Service",
"operation":"Import",
"sequenceNumber":"2",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-23 07:20:16",
"endTime":"2018-07-23 07:20:32",
"resourceType":"DB Util"
}
]
},
{
"id":"88-1V5Y6J",
"planName":"Data Mig",
"description":"Data Mig",
"status":"Success",
"startDate":"2018-07-23 06:20:12",
"endDate":"2018-07-23 06:20:58",
"source":"Dev",
"target":"Prod",
"packageName":null,
"resources":[
{
```

```
"id":"88-1V5Y6K",
"name":"Application Data Service",
"operation":"Export",
"sequenceNumber":"1",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-23 06:20:12",
"endTime":"2018-07-23 06:20:27",
"resourceType":"DB Util"
},
{
"id":"88-1V5Y6L",
"name":"Application Data Service",
"operation":"Import",
"sequenceNumber":"2",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-23 06:20:43",
"endTime":"2018-07-23 06:20:58",
"resourceType":"DB Util"
}
]
},
{
"id":"88-1V5Y5Y",
"planName":"Data Mig",
"description":"Data Mig",
"status":"Success",
"startDate":"2018-07-23 06:11:24",
"endDate":"2018-07-23 06:12:37",
"source":"Dev",
"target":"Prod",
"packageName":null,
"resources":[
{
"id":"88-1V5Y5Z",
"name":"Application Data Service",
"operation":"Export",
"sequenceNumber":"1",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-23 06:11:24",
"endTime":"2018-07-23 06:11:39",
"resourceType":"DB Util"
},
{
"id":"88-1V5Y60",
"name":"Application Data Service",
"operation":"Import",
"sequenceNumber":"2",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-23 06:11:54",
"endTime":"2018-07-23 06:12:37",
"resourceType":"DB Util"
}
]
},
{
"id":"88-1V5Y5S",
"planName":"Data Mig",
"description":"Data Mig",
"status":"Success",
"startDate":"2018-07-23 06:11:13",
"endDate":"2018-07-23 06:12:17",
"source":"Dev",
"target":"Prod",
"packageName":null,
"resources":[
{
"id":"88-1V5Y5T",
"name":"Application Data Service",
"operation":"Export",
"sequenceNumber":"1",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-23 06:11:13",
"endTime":"2018-07-23 06:11:28",
"resourceType":"DB Util"
},
{
"id":"88-1V5Y5U",
"name":"Application Data Service",
"operation":"Import",
"sequenceNumber":"2",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-23 06:11:43",
"endTime":"2018-07-23 06:12:17",
"resourceType":"DB Util"
}
]
},
{
"id":"88-1V5Y5M",
"planName":"Data Mig",
"description":"Data Mig",
"status":"Success",
"startDate":"2018-07-23 06:11:07",
"endDate":"2018-07-23 06:11:53",
"source":"Dev",
"target":"Prod",
"packageName":null,
"resources":[
{
"id":"88-1V5Y5N",
"name":"Application Data Service",
"operation":"Export",
"sequenceNumber":"1",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-23 06:11:07",
"endTime":"2018-07-23 06:11:22",
"resourceType":"DB Util"
},
{
"id":"88-1V5Y5O",
"name":"Application Data Service",
"operation":"Import",
"sequenceNumber":"2",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-23 06:11:38",
"endTime":"2018-07-23 06:11:53",
"resourceType":"DB Util"
}
]
},
{
"id":"88-1V5Y5P",
"planName":"Data Mig",
"description":"Data Mig",
"status":"Error",
"startDate":"2018-07-23 06:11:09",
"endDate":"2018-07-23 06:11:39",
"source":"Dev",
"target":"Prod",
"packageName":null,
"resources":[
{
"id":"88-1V5Y5Q",
"name":"Application Data Service",
"operation":"Export",
"sequenceNumber":"1",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-23 06:11:09",
"endTime":"2018-07-23 06:11:24",
"resourceType":"DB Util"
},
{
"id":"88-1V5Y5R",
"name":"Application Data Service",
"operation":"Import",
"sequenceNumber":"2",
"mode":"Asynchronous",
"status":"Not Started",
"startTime":"",
"endTime":"",
"resourceType":"DB Util"
}
]
},
{
"id":"88-1V5Y4X",
"planName":"Data Mig",
"description":"Data Mig",
"status":"Success",
"startDate":"2018-07-23 02:47:14",
"endDate":"2018-07-23 02:48:00",
"source":"Dev",
"target":"Prod",
"packageName":null,
"resources":[
{
"id":"88-1V5Y4Y",
"name":"Application Data Service",
"operation":"Export",
"sequenceNumber":"1",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-23 02:47:14",
"endTime":"2018-07-23 02:47:29",
"resourceType":"DB Util"
},
{
"id":"88-1V5Y4Z",
"name":"Application Data Service",
"operation":"Import",
"sequenceNumber":"2",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-23 02:47:44",
"endTime":"2018-07-23 02:47:59",
"resourceType":"DB Util"
}
]
},
{
"id":"88-1V5Y4B",
"planName":"Data Mig",
"description":"Data Mig",
"status":"Success",
"startDate":"2018-07-23 02:36:26",
"endDate":"2018-07-23 02:38:36",
"source":"Dev",
"target":"Prod",
"packageName":null,
"resources":[
{
"id":"88-1V5Y4C",
"name":"Application Data Service",
"operation":"Export",
"sequenceNumber":"1",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-23 02:37:50",
"endTime":"2018-07-23 02:38:06",
"resourceType":"DB Util"
},
{
"id":"88-1V5Y4D",
"name":"Application Data Service",
"operation":"Import",
"sequenceNumber":"2",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-23 02:38:21",
"endTime":"2018-07-23 02:38:36",
"resourceType":"DB Util"
}
]
},
{
"id":"88-1V5Y48",
"planName":"Data Mig",
"description":"Data Mig",
"status":"Success",
"startDate":"2018-07-23 02:30:33",
"endDate":"2018-07-23 02:31:19",
"source":"Dev",
"target":"Prod",
"packageName":null,
"resources":[
{
"id":"88-1V5Y49",
"name":"Application Data Service",
"operation":"Export",
"sequenceNumber":"1",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-23 02:30:33",
"endTime":"2018-07-23 02:30:48",
"resourceType":"DB Util"
},
{
"id":"88-1V5Y4A",
"name":"Application Data Service",
"operation":"Import",
"sequenceNumber":"2",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-23 02:31:03",
"endTime":"2018-07-23 02:31:18",
"resourceType":"DB Util"
}
]
},
{
"id":"88-1V5Y3J",
"planName":"Data Mig",
"description":"Data Mig",
"status":"Success",
"startDate":"2018-07-22 11:26:29",
"endDate":"2018-07-22 11:27:15",
"source":"Dev",
"target":"Prod",
"packageName":"",
"resources":[
{
"id":"88-1V5Y3K",
"name":"Application Data Service",
"operation":"Export",
"sequenceNumber":"1",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-22 11:26:29",
"endTime":"2018-07-22 11:26:44",
"resourceType":"DB Util"
},
{
"id":"88-1V5Y3L",
"name":"Application Data Service",
"operation":"Import",
"sequenceNumber":"2",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-22 11:27:00",
"endTime":"2018-07-22 11:27:15",
"resourceType":"DB Util"
}
]
},
{
"id":"88-1V5Y35",
"planName":"Data Mig",
"description":"Data Mig",
"status":"Success",
"startDate":"2018-07-22 11:17:16",
"endDate":"2018-07-22 11:18:03",
"source":"Dev",
"target":"Prod",
"packageName":"",
"resources":[
{
"id":"88-1V5Y36",
"name":"Application Data Service",
"operation":"Export",
"sequenceNumber":"1",
"mode":"Asynchronous",
"status":"Success",
"startTime":"2018-07-22 11:17:16",
"endTime":"2018-07-22 11:17:31",
"resourceType":"DB Util"
},
{
"id":"88-1V5Y37",
"name":"Application Data Service",
"operation":"Import",
"sequenceNumber":"2",
        "mode":"Asynchronous",
        "status":"Success",
        "startTime":"2018-07-22 11:17:48",
        "endTime":"2018-07-22 11:18:03",
        "resourceType":"DB Util"
        }
        ]
        }
       ]
```

Getting Migration History by ID Number

You can get a migration history by ID number for your migration by sending an HTTP GET request to the Siebel Migration Application.

The following details are for a request to get a migration history by ID number for a migration:

- **URI:** `https://{hostname}:{port}/siebel/v1.0/migration/history/{id}`
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request Body:** None

The following are the details for the response to a successful request:

- **HTTP Code**: 200
- **Content-Type:** application/json
- **Response Body:**

```
{
 "id":"88-1V5WON",
 "planName":"IRRMigration",
 "description":"IRRMigration",
 "status":"Error",
 "startDate":"2018-07-17 04:28:06",
 "endDate":"2018-07-17 04:28:22",
 "source":"Dev",
 "target":"Prod1",
 "packageName":"",
 "resources":[
 {
 "id":"88-1V5WOO",
 "name":"Schema Service",
 "operation":"Export",
 "sequenceNumber":"1",
 "mode":"Asynchronous",
 "status":"Not Applicable",
 "startTime":"",
 "endTime":"",
 "resourceType":"DB Util"
 },
 {
 "id":"88-1V5WOP",
 "name":"Schema Service",
 "operation":"Import",
 "sequenceNumber":"2",
 "mode":"Asynchronous",
 "status":"Not Applicable",
 "startTime":"",
 "endTime":"",
 "resourceType":"DB Util"
 },
 {
 "id":"88-1V5WOQ",
 "name":"Incremental Runtime Repository Data Service",
 "operation":"Export",
 "sequenceNumber":"3",
 "mode":"Asynchronous",
 "status":"Error",
 "startTime":"2018-07-17 04:28:07",
 "endTime":"2018-07-17 04:28:22",
 "resourceType":"DB Util"
 },
 {
 "id":"88-1V5WOR",
 "name":"Incremental Runtime Repository Data Service",
 "operation":"Import",
 "sequenceNumber":"4",
 "mode":"Asynchronous",
 "status":"Not Started",
 "startTime":"",
 "endTime":"",
 "resourceType":"DB Util"
 },
 {
 "id":"88-1V5WOS",
 "name":"Incremental Runtime Repository Data Service",
 "sequenceNumber":"5",
 "mode":"Asynchronous",
 "status":"Not Started",
 "startTime":"",
 "endTime":"",
 "resourceType":"DB Util"
 },
 {
 "id":"88-1V5WOT",
 "name":"Incremental Runtime Repository Data Service",
 "operation":"DBCheck",
 "sequenceNumber":"6",
 "mode":"Asynchronous",
 "status":"Not Started",
 "startTime":"",
 "endTime":"",
 "resourceType":"DB Util"
 }
 ]
}
```

Getting Migration History by ID and Resource Name

You can get the migration history by ID and resource name for your migration by sending an HTTP Get request to the Siebel Migration Application.

The following details are for a request to get the migration history by ID and resource name for a migration:

- **URI:** `https://{hostname}:{port}/siebel/v1.0/migration/history/{id}/{resourceName}`
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request Body:** None

The following are the details for the response to a successful request:

- **HTTP Code:** 200

- **Content-Type:** application/json
- **Response Body:**

```
{
 "id":"88-1V5WON",
 "planName":"IRRMigration",
 "description":"IRRMigration",
 "status":"Error",
 "startDate":"2018-07-17 04:28:06",
 "endDate":"2018-07-17 04:28:22",
 "source":"Dev",
 "target":"Prod1",
 "packageName":"",
 "resources":[
 {
 "id":"88-1V5WOQ",
 "name":"Incremental Runtime Repository Data Service",
 "operation":"Export",
 "sequenceNumber":"3",
 "mode":"Asynchronous",
 "status":"Error",
 "startTime":"2018-07-17 04:28:07",
 "endTime":"2018-07-17 04:28:22",
 "resourceType":"DB Util"
 },
 {
 "id":"88-1V5WOR",
 "name":"Incremental Runtime Repository Data Service",
 "operation":"Import",
 "sequenceNumber":"4",
 "mode":"Asynchronous",
 "status":"Not Started",
 "startTime":"",
 "endTime":"",
 "resourceType":"DB Util"
 },
 {
 "id":"88-1V5WOS",
 "name":"Incremental Runtime Repository Data Service",
 "sequenceNumber":"5",
 "mode":"Asynchronous",
 "status":"Not Started",
 "startTime":"",
 "endTime":"",
 "resourceType":"DB Util"
 },
 {
 "id":"88-1V5WOT",
 "name":"Incremental Runtime Repository Data Service",
 "operation":"DBCheck",
 "sequenceNumber":"6",
 "mode":"Asynchronous",
 "status":"Not Started",
 "startTime":"",
 "endTime":"",
 "resourceType":"DB Util"
 }
 ]
}
```

Getting Migration History by ID, Resource Name, and Operation

You can get migration history by ID, resource name, and operation for your migration by sending an HTTP GET request to the Siebel Migration Application.

The following details are for a request to get migration history by ID, resource name, and operation for a migration:

- **URI:** `https://{hostname}:{port}/siebel/v1.0/migration/history/{id}/{resourceName}/{operation}`
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request Body:** None

The following are the details for the response to a successful request:

- **HTTP Code:** 200
- **Content-Type:** application/json
- **Response Body:**

```
{
 "id":"88-1V5WON",
 "planName":"IRRMigration",
 "description":"IRRMigration",
 "status":"Error",
 "startDate":"2018-07-17 04:28:06",
 "endDate":"2018-07-17 04:28:22",
 "source":"Dev",
 "target":"Prod1",
 "packageName":"",
 "resources":[
 {
 "id":"88-1V5WOQ",
 "name":"Incremental Runtime Repository Data Service",
 "operation":"Export",
 "sequenceNumber":"3",
 "mode":"Asynchronous",
 "status":"Error",
 "startTime":"2018-07-17 04:28:07",
 "endTime":"2018-07-17 04:28:22",
 "resourceType":"DB Util"
 }
 ]
}
```

Getting the Migration History Operation Log

You can get the migration history operation log for your migration by sending an HTTP GET request to the Siebel Migration Application.

The following details are for a request to get the migration history operation log for a migration:

- **URI:** `https://{hostname}:{port}/siebel/v1.0/migration/history/{id}/{resourceName}/{operation}/log`
- **HTTP Method:** GET
- **Content-Type:** application/json
- **Authorization:** Basic
- **Request Body:** None

The response returns the migration history operation log for your migration. The following are the details for the response to a successful request:

- **HTTP Code:** 200

- **Content-Type:** application/json

25 Implementing Siebel High-Availability Upgrade Using Oracle Golden Gate

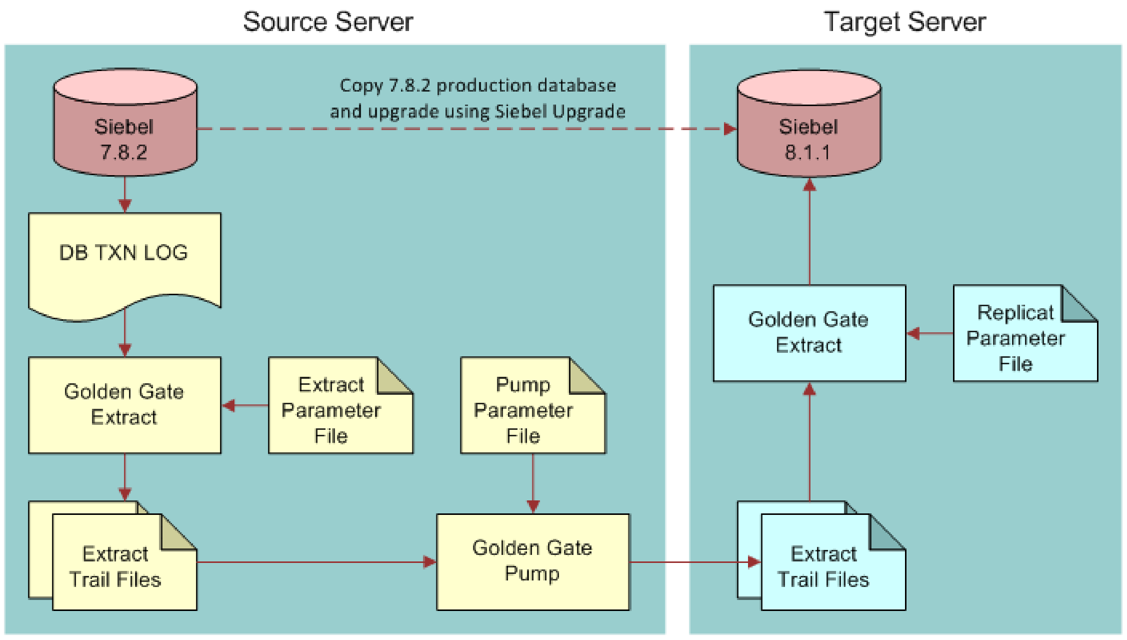
Implementing Siebel High-Availability Upgrade Using Oracle Golden Gate

This chapter describes how to implement Siebel High-Availability (HA) upgrade using Oracle GoldenGate to upgrade to the current release. It includes the following topics:

- *Overview of Oracle Golden Gate Implementation*
- *About Siebel High Availability Upgrade Files*
- *Extracting Oracle Golden Gate Files*
- *Example of Implementing a Siebel High-Availability Upgrade*
- *Limitations When Performing a Siebel Upgrade with Oracle Golden Gate*

Overview of Oracle GoldenGate Implementation

You use the standard Siebel upgrade process detailed in this guide to upgrade your Siebel database from one supported Siebel CRM version to another version. The duration of the upgrade process and the potential for having your live environment unavailable during the process depend on the database size and other factors. Performing a high-availability upgrade using Oracle GoldenGate enables you to keep your Siebel CRM production environment available to your customers during the upgrade process while also causing the least interruption to your business users because you do not have to shut down your Siebel environment during the upgrade. The following figure provides an overview of the Oracle GoldenGate architecture.



This figure shows two servers (source server and target server) with Oracle GoldenGate software installed on each of them. Each server has a database instance: one is the source system Siebel CRM version 7.8.2, and the other has the target system, the current release of Siebel CRM. The directory structure and files reflect this environment.

For example, other upgrade situations might involve a single server hosting both databases (either in the same instance or in two different instances) with one instance of Oracle GoldenGate software. In this situation, the Oracle GoldenGate pump process and its parameter files are not required. The extract files created by the Oracle GoldenGate Extract process can be used by the Replicat process. In this case, the file prefix value in the replicat OBEY file (such as r782_811.oby) must be changed to the one used in the Extract process.

For more information on Oracle GoldenGate, see Oracle Technology Network (http://www.oracle.com/technetwork/indexes/documentation/index.html).

About Siebel High Availability Upgrade Files

Oracle GoldenGate software is implemented with a set of files that enable you to decrease the downtime of your Siebel production database during an upgrade. These files are provided in the Siebel Database Server installation directory:

```
SIEBEL_ROOT/dbsrvr/COMMON
```

These files are compressed with names like the following, depending on the database platform:

- GoldenGateSIA782_SIA811_HA_Upg_ORA.zip
- GoldenGateSIA782_SIA811_HA_Upg_ORA.tar.gz
- GoldenGateSIA782_SIA811_HA_Upg_DB2UDB.tar.gz
- GoldenGateSIA782_SIA811_HA_Upg_DB2UDB.tar.gz
- GoldenGateSIA782_SIA811_HA_Upg_MSSQL.tar.gz
- GoldenGateSIA782_SIA811_HA_Upg_MSSQL.tar.gz

Note: Oracle Siebel CRM provides versions 7.8.2 to 8.1.1 of the Golden Gate upgrade scripts that are ready to use. These scripts are provided as references only. For customers who plan to use the Oracle Golden Gate solution to achieve Siebel HA upgrade, please contact Oracle Advanced Customer Services (ACS). They can extend the scripts based on the actual upgrade paths and help customers implement the HA upgrade solution.

Extracting Oracle GoldenGate Files

The ZIP or tar files include a number parameter files and OBEY files, which can be used with Oracle GoldenGate software. For more information, see *About Siebel High Availability Upgrade Files*.

Note: Some of the files are specific to a database platform. Depending on your database platform, database-specific ZIP or tar files must be used to extract the Oracle GoldenGate parameter files.

To extract compressed Oracle Golden Gate files

1. From your `dbsrvr` directory, extract the files from the following ZIP or tar file:

   ```
   GoldenGateSIA782_SIA811_HA_Upg_*.*
   ```

2. View the following files after extracting them:

   `SIA8119/dbsrvr/COMMON`

   - GoldenGateSIA782_SIA811_HA_Upg_ORA.zip (Windows only)
   - GoldenGateSIA782_SIA811_HA_Upg_DB2UDB.zip (Windows only)
   - GoldenGateSIA782_SIA811_HA_Upg_MSSQL.zip (Windows only)
   - GoldenGateSIA782_SIA811_HA_Upg_ORA.tar.gz (UNIX and Linux only)
   - GoldenGateSIA782_SIA811_HA_Upg_DB2UDB.tar.gz (UNIX and Linux only)
   - GoldenGateSIA782_SIA811_HA_Upg_MSSQL.tar.gz (UNIX and Linux only)

   `SIA8119/dbsrvr/GoldenGate/SIA782_811/SourceServer/GLOBALS`

   - dirprm

     d_782sia.prm

     e_782sia.oby

     e_782sia.prm

     mgr.prm

     p_782sia.oby

     p_782sia.prm

     s782sia_trandata.oby

     siebelMacro.mac
   - dirrpt

     extractReports

     pumpReports

   `SIA8119/dbsrvr/GoldenGate/SIA782_811/TargetServer/GLOBALS`

   - dirprm

     mgr.prm

     r782_811.oby

     r782_811.prm

     siebelMacro.mac
   - dirrpt

     replicatReports

3. Assuming a directory structure illustrated in step 2`.../SIA8119/dbsrvr/COMMON`, run tar or extract the files to the following directory, depending on your operating system:

   - **Windows.** `cd.../SIA8119/dbsrvr`
   - **UNIX.** `tar -xvf ./COMMON/GoldenGateSIA782_SIA811_HA_Upg_ORA.tar.gz`

   See the following table for a description of the source server files and directories.

File or Directory Name	Description	Required or Optional Changes
GLOBALS	Global Parameters file for the Oracle GoldenGate processes. This is an empty file.	You can add the required parameters for your environment.
d_782sia.prm	Parameters file for the Oracle GoldenGate executable **defgen**. This file contains the list of Siebel CRM version 7.8.2 tables from which data is extracted from the Siebel CRM version 7.8.2 database log file by Oracle GoldenGate.	You can do the following: • Change the login information for the source database. • Change the table owner and schema name for the tables. • Add your custom tables (X_* tables).
e_782sia.oby	The OBEY file used to create the Oracle GoldenGate Extract process.	You can do the following: • (Optional) Extract the file size. • (Optional) Add **start** command.
e_782sia.prm	The parameter file for the Oracle GoldenGate Extract process. This file contains the list of Siebel 7.8.2 application tables from which data is extracted from the Siebel 7.8.2 database log file by Oracle GoldenGate. The tables include the following: • Siebel Enterprise Integration Manager tables • Siebel repository tables • Other tables related to Siebel Remote log files and other similar files	You can do the following: • (Optional) Report options. • Add custom tables designated by the (X_*) prefix.
mgr.prm	The parameter file for the Oracle GoldenGate Manager process. The same file is included in both source and target **dirprm** directories.	You can do the following: • (Optional) Port and Port List numbers. • (Optional) other parameters.
p_782sia.oby	The OBEY file used to create the Oracle GoldenGate pump process.	You can do the following: • (Optional) pump file size. • (Optional) Add the **start** command.

File or Directory Name	Description	Required or Optional Changes
p_782sia.prm	The parameter file for the Oracle GoldenGate pump process.	No changes required.
s782sia_trandata.oby	The OBEY file that contains the Oracle GoldenGate **`ADD TRANDATA`** command. Use this command to get a list of the tables whose data will be extracted from the Siebel database 7.8.2. This command also requests supplemental log information for the tables to be executed by the database server. For some tables, additional columns are included because those columns are required to populate the tables in the upgraded target database.	You can do the following: • Change the login information for the source database. • Change the table owner and schema name for the tables. • Add custom tables added by the customer (X_* tables).
siebelMacro.mac	Contains the Oracle GoldenGate macros that are used to define the following: • Source database login information • Target database login information • Source database table owner and schema name • Target database table owner and schema name • Target server name • Target manager port number • Primary language for the source Siebel database These macros are used in the parameter files for the Extract process, pump, and Replicat process. For more information, see *Oracle GoldenGate Macros*.	
extractReports	The directory where the report files for the Extract process are stored.	Ensure that this directory is created in the **`dirrpt`** directory of Oracle GoldenGate on the source server.
pumpReports	The directory where the report files for the pump process are stored.	Ensure that this directory is created in the **`dirrpt`** directory of Oracle GoldenGate on the source server.

See the following table for a description of some of the target server files and directories.

File or Directory Name	Description or Usage	Changes to Make
GLOBALS	Global parameters file for the Oracle GoldenGate processes. This file has only one line, which is required: checkpoint table GGS_ CHECKPOINT.	• (Optional) Add the required parameters for your environment.

File or Directory Name	Description or Usage	Changes to Make
		• Change the checkpoint table name if you have a defined checkpoint table with any other name. • Add a checkpoint table in the target database by using the command: **`ADD CHECKPOINTTABLE table_owner_name.GGS_CHECKPOINT`**
mgr.prm	The parameters file for the Oracle GoldenGate Manager process. This file is included in both the source and the target **`dirprm`** directories.	• (Optional) provide the port and the post list numbers. • (Optional) provide the other required parameters.
r782_811.oby	The OBEY file to create the Oracle GoldenGate Replicat process.	(Optional) Add the **`start`** command.
r782_811.prm	This file contains the list of Siebel SIA 8.1.1 tables for which the transaction data that was extracted from Siebel 7.8.2 SIA database is replicated in the target Siebel 8.1.1 SIA database, applying the required upgrade logic to the table.	• (Optional) Add the report options. • Add any additional upgrade logic for Siebel database tables and custom tables (X_*).
siebelMacro.mac	Contains the Oracle GoldenGate macros that define following: • Source database login information • Target database login information • Source database table owner and schema name • Target database table owner and schema name • Target server name, • Target manager port number • Primary language for the source Siebel database This file has the same values as the file on the source server. See the previous table for more information. These macros are used in the parameter files for the Extract, pump, and Replicat process.	You can copy the file from the instance of Oracle GoldenGate on the source database after the macro values have been changed for this file in the source server.
replicatReports	The directory where the report files for the Replicat process are stored.	Ensure that this directory is created in the Oracle GoldenGate **`dirrpt`** directory on the target server.

Note: The standard Siebel upgrade process, described in this guide, does not use the ZIP files, tar files, or the extracted files and directories. Oracle GoldenGate software does not use the ZIP files, tar files, extracted files, and directories directly. These files must be located in the **`dirprm`** directory for the Oracle GoldenGate software installation.

The extracted files must be copied to the `dirprm` directory in the Oracle GoldenGate software installation and modified to reflect your environment. For example, modify the following: database login and password information, table owner and schema names, port numbers and the target server name and so on. Make sure that the directories mentioned in this topic have been created under the `dirrpt` directory where Oracle GoldenGate software has been installed.

Oracle GoldenGate Macros

The Oracle GoldenGate macros identify the following:

- Source database table owner and schema name.
- Target database table owner and schema name.
- Source database login and password information.
- Target database login and password information.
- The remote host name. In other words, the server name where the Oracle GoldenGate instance for the target database is located.
- The port number used by the Oracle GoldenGate Manager process for the Oracle GoldenGate instance in the target database.
- Optionally, the macro determines the string value for the DB_LAST_UPD_SRC column in the target database.
- The language code for the primary language of your Siebel CRM installation on the source database.

Example of Implementing a Siebel High-Availability Upgrade

This topic provides an example of how to implement a Siebel high-availability upgrade, using Oracle GoldenGate software:

1. Install Oracle GoldenGate software on the source and target servers, and configure them.

   For more information on Oracle GoldenGate, see Oracle Technology Network (http://www.oracle.com/technetwork/indexes/documentation/index.html).
2. Install the current release of Siebel CRM, including the Siebel Server, Siebel Database Server, and Siebel Gateway.
3. From the installation of the current release of Siebel CRM, extract the Oracle GoldenGate files from the installed ZIP or tar file as described in *Extracting Oracle Golden Gate Files*, and copy the extracted files to the `dirprm` directory of the Oracle GoldenGate installation.
4. Make all the required changes to those files. Provide database login and password information, as well as table owner, target server name, and so on. Create the required directories under the `dirrpt` directory.
5. Customize the Oracle GoldenGate parameter files to include any custom tables that you have previously created (X_*), including any logic required for those tables.
6. Using the Oracle GoldenGate executable `defgen` and the modified parameter file d_782sia.prm, generate the schema definitions for Siebel 7.8.2 tables.

   This parameter file defines the output file as Siebel782.dat. This file is generated under the `dirdat` directory. Copy this file to the target server's `dirdat` directory.
7. Create a new database instance on the target server to host the Siebel 7.8.2 database that you will upgrade.
8. Ensure that all disconnected users synchronize their local database with the server database.

9. Bring the production database to an idle state by disconnecting all users and stopping all Siebel Server processes.
10. Take a full dump of the database. Using the dump file, create a new copy of the database in the target server.

    This database will be upgraded to the current release of Siebel CRM.
11. Start the Oracle GoldenGate Extract and pump process on the source server to capture all new transactions in the Siebel 7.8.2 database. Start the Siebel application and allow users to connect and continue to use the application.
12. Ensure that the Oracle GoldenGate processes are running and extract files are created on the source server and are created in the target server by the Oracle GoldenGate pump process.
13. Start the standard Siebel upgrade process on the target server to upgrade the Siebel 7.8.2 database and Siebel Business Applications to the current release of Siebel CRM.
14. Monitor the Oracle GoldenGate Extract process and pump process to ensure that they are capturing data from the source server. Move the data to the target server continuously.
15. After the upgrade to Siebel 8.1.1.x is complete and the upgrade is validated, start the Oracle GoldenGate Replicat process on the target server to apply the captured transactions in the extract files to the upgraded database for the current release of Siebel CRM.
16. After the Oracle GoldenGate Replicat process has retrieved all the transactions in the source Siebel 7.8.2 database, all disconnected users must synchronize their local database with the server database and stop using the Siebel application in disconnected mode.
17. Bring the production database to an idle state again, by disconnecting all users and stopping all Siebel Server processes.
18. Check that the last few identified transactions on the source database are reflected in the upgraded database for the current release of Siebel CRM. Start all the required Siebel server processes and start the Siebel application. The upgraded Siebel CRM database and Siebel applications are now ready for use.
19. Migrate all users so that they can start using the new upgraded database and Siebel applications for the current release of Siebel CRM. Shut down the Siebel 7.8.2 database, and stop the Oracle GoldenGate Extract, pump, and Replicat processes.
20. If some users are disconnected (Siebel Mobile Web Client users), then create a new version of the local database for these users, for the current release of Siebel CRM. After those users initialize their local database with the new local database extract, they can start using the application in disconnected mode.

    The upgrade of Siebel 7.8.2.x to the current release of Siebel CRM, using Oracle GoldenGate with minimum downtime of the Siebel application (version 7.8.2), is now complete.

Limitations When Performing a Siebel Upgrade with Oracle GoldenGate

The following limitations apply when performing a Siebel upgrade with Oracle GoldenGate software:

- **You must not perform any of the following operations in the Siebel 7.8.2 database after the target database has been created when preparing to upgrade to the current release:**
    - Creating or modifying schemas, business components, applets, views, and so on in the Siebel Repository
    - Creating or modifying workflows, assignment rules, audit rules, and so on
- **Siebel Enterprise Integration Manager tables are not replicated.** This limitation is because all Siebel Enterprise Integration Manager operations on the source database result in changes to the application tables, and those transactions from the application tables are replicated.

- **Transactions in the docking transactions log tables are not replicated.** This limitation is because disconnected users are expected to get a newly installed local database from the upgraded Siebel database following the completed upgrade when Oracle GoldenGate has applied all the Siebel CRM version 7.8.2 transactions to the database for the current release.
- **Log files and tables with data that are temporary are not replicated.**
- **For LOV-based column values that are used in the upgrade steps, the Oracle GoldenGate Replicat parameter file contains values in English (ENU) from the S_LST_OF_VAL table.** For customers whose primary language is not English, some of these steps must be changed to use values in the customer's installed language.
- **Using the OBEY files to define the Extract, pump, and Replicat processes and using the GGSCI command does not start those processes.** You must run the `start` command from within the Oracle GoldenGate secure command interface (GGSCI) when you are ready to start the processes.

26 Overview of Performing a Siebel Database Upgrade

Overview of Performing a Siebel Database Upgrade

This chapter provides process information on how to upgrade your Siebel database. It includes the following topics:

- *Roadmap for Siebel Database Environment Upgrade*
- *Process of Upgrading a Siebel Development Environment*
- *Process of Upgrading a Siebel Production Test Environment*
- *Process of Tuning Siebel Upgrade Performance*
- *Process of Upgrading a Siebel Production Environment*

Roadmap for Siebel Database Environment Upgrade

This roadmap provides an overview of tasks required to upgrade your Siebel environment. Use this roadmap to determine the applicability of each task to each environment. For example, if a task has *No* listed in the Prod column, then that task is not required for production test, or production environment upgrades. All tasks required for the production test upgrade are also required for the production environment upgrade. The development environment has more required tasks. Print and review this roadmap as you plan and work through your upgrade.

Note: Links to other topics in this guide are provided throughout this chapter. These links provide additional information that cannot be provided in these brief overviews. You must, however, perform the upgrade in the proper order. For information on general planning, see this chapter as well as *Siebel Database Upgrade Planning* and *Application Planning for a Siebel Upgrade* To begin the process of upgrading your Siebel database, see *Preparing for Siebel Database Upgrade* then refer to each subsequent chapter.

Information about the following is included in this roadmap:

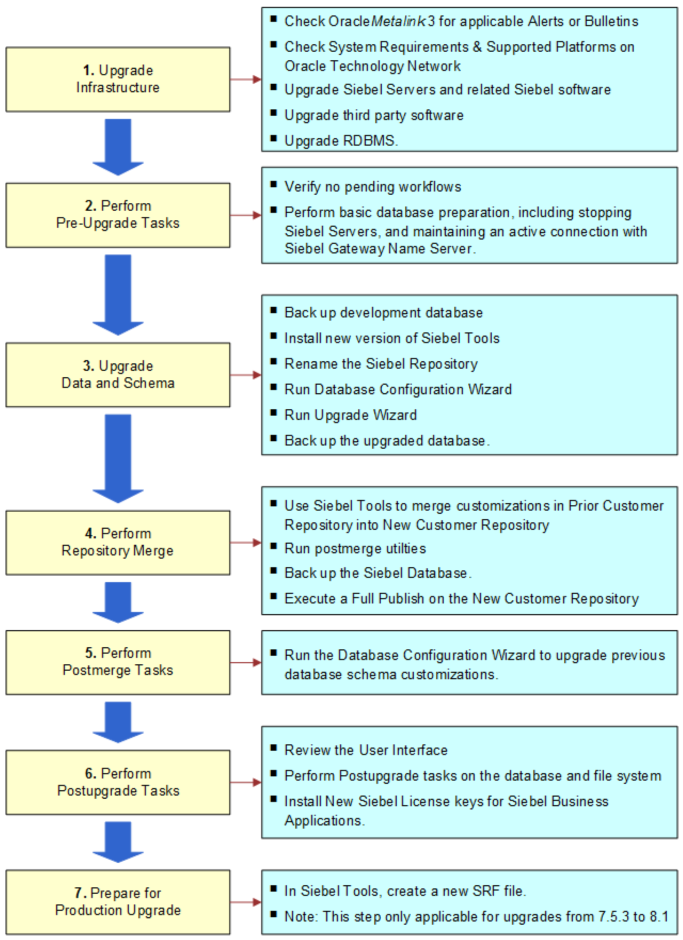
- *Evaluating the Environment*
- *Planning the Upgrade*
- *Configuring the Environment*
- *Testing the Environment*
- *Implementing the Environment*

The basic tasks for upgrading Siebel database environments, as shown in the following image, are as follows:

1. Upgrade infrastructure:
   - Check for applicable alerts or bulletins.
   - Check System Requirements and Supported Platforms on Oracle Technology Network.
   - Upgrade Siebel Servers and related Siebel software.

   - Upgrade RDBMS.
2. Perform pre-upgrade tasks:
   - Verify that there are no pending workflows.
   - Perform basic database preparation, including stopping Siebel Servers, and maintaining an active connection with Siebel Gateway Name Server.
3. Upgrade data and schema:
   - Back up the development database.
   - Install new version of Siebel Tools.
   - Rename the Siebel Repository.
   - Run the Database Configuration Wizard.
   - Run the Upgrade Wizard.
   - Back up the upgraded database.
4. Perform repository merge:
   - Use Siebel Tools to merge customizations in Prior Customer Repository into New Customer Repository.
   - Run postmerge utilities.
   - Back up the Siebel database.
   - Execute a Full Publish on the New Customer Repository.
5. Perform postmerge tasks:
   - Run the Database Configuration Wizard to upgrade previous database schema customizations.
6. Perform postupgrade tasks:
   - Review the user interface.
   - Perform postupgrade tasks on the database and file system.
   - Install new Siebel license keys for Siebel Business Applications.
7. Prepare for production upgrade:
   - Create a new SRF file in Siebel Tools.

     This step applies only for upgrades from version 7.5.3 to 8.1.


1. Upgrade Infrastructure
Check Oracle Metalink3 for applicable Alerts or Bulletins
Check System Requirements & Supported Platforms on Oracle Technology Network
Upgrade Siebel Servers and related Siebel software
Upgrade third party software
Upgrade RDBMS.
2. Perform Pre-Upgrade Tasks
Verify no pending workflows
Perform basic database preparation, including stopping Siebel Servers, and maintaining an active connection with Siebel Gateway Name Server.
3. Upgrade Data and Schema
Back up development database
Install new version of Siebel Tools
Rename the Siebel Repository
Run Database Configuration Wizard
Run Upgrade Wizard
Back up the upgraded database.
4. Perform Repository Merge
Use Siebel Tools to merge customizations in Prior Customer Repository into New Customer Repository
Run postmerge utilties
Back up the Siebel Database.
Execute a Full Publish on the New Customer Repository
5. Perform Postmerge Tasks
Run the Database Configuration Wizard to upgrade previous database schema customizations.
6. Perform Postupgrade Tasks
Review the User Interface
Perform Postupgrade tasks on the database and file system
Install New Siebel License keys for Siebel Business Applications.
7. Prepare for Production Upgrade
In Siebel Tools, create a new SRF file.
Note: This step only applicable for upgrades from 7.5.3 to 8.1

For a more detailed view of the upgrade steps for each Siebel database environment, see the following topics:

- *Process of Upgrading a Siebel Development Environment*
- *Process of Upgrading a Siebel Production Test Environment*
- *Process of Tuning Siebel Upgrade Performance*
- *Process of Upgrading a Siebel Production Environment*

Evaluating the Environment

Whether you are beginning your upgrade with the development environment or the production test environment, it is advisable to begin by evaluating the process to determine resource allocation and other concerns. The following table includes information relevant to this phase of the upgrade.

Upgrade Tasks	Where to Find Documentation	Dev?	Prod?
1. Check My Oracle Support for recently published articles regarding your upgrade.	My Oracle Support (http:// support.oracle.com)	Yes	Yes
2. Check *Siebel Release Notes* on My Oracle Support for a list of known issues.	My Oracle Support (http:// support.oracle.com), searching for Siebel CRM update guides and release notes,	Yes	Yes

Upgrade Tasks	Where to Find Documentation	Dev?	Prod?
	or the readme.html file provided with the installation media for each release.		
3. Review hardware and third-party software requirements.	The Certifications tab on My Oracle Support.	Yes	Yes
4. If you need help evaluating your environment, then contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.	Contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services	Yes	Yes

Planning the Upgrade

Review the *Siebel Database Upgrade Guide* to plan and further determine the scope of the upgrade process. The following table lists chapters in the *Siebel Database Upgrade Guide* to consider during the planning phase.

Upgrade Tasks	Where to Find Documentation	Dev?	Prod?
1. Review the *Siebel Database Upgrade Guide* for new features in the current release.	*What's New in This Release*	Yes	Yes
2. Review how to perform the Siebel database upgrade and how the upgrade works step-by-step.	*Overview of Siebel Database Environments* and *Siebel Database Upgrade Planning*	Yes	Yes
3. Review Siebel database upgrade planning and about upgrade considerations for your Siebel user interface.	*Siebel Database Upgrade Planning*	Yes	Yes
4. Review planning for Siebel Business Applications prior to upgrade.	*Application Planning for a Siebel Upgrade*	Yes	Yes

Configuring the Environment

This is the most time-consuming and resource-intensive phase of the Siebel database upgrade. Many tasks in this phase are performed only during a development environment upgrade. Perform these tasks only if you are upgrading a development environment. All other tasks are performed during all upgrades. The following table includes information relevant to this phase of the upgrade.

Upgrade Tasks	Where to Find Documentation	Dev?	Prod
1. Copy the ancestor repositories.	For details, see *Ancestor Repositories*.	Yes	No

Upgrade Tasks	Where to Find Documentation	Dev?	Prod
2. Upgrade the servers. Verify that you have met all requirements for the upgrade. You might be required to install the Siebel Gateway, Siebel Servers, and Siebel Application Interface.	For more information on server upgrade and installation, see the *Siebel Installation Guide* . See also the Certifications tab on My Oracle Support.	Yes	Yes
3. Upgrade third-party software. You might need, for example, to upgrade operating system software.	See the Certifications tab on My Oracle Support.	Yes	Yes
4. Upgrade the RDBMS.	See the Certifications tab on My Oracle Support. For details on upgrading the RDBMS, see *About Upgrading Your RDBMS in the Siebel Environment*.	Yes	Yes
5. Perform any preupgrade tasks on your specific database.	For details on IBM DB2.see *Perform Preupgrade Tasks for IBM DB2*. For details on Oracle Database, see *Perform Preupgrade Tasks for Oracle Database*. For details on Microsoft SQL Server, see *Perform Preupgrade Tasks for Microsoft SQL Server*.	Yes	Yes
6. Preparing the Siebel database for an upgrade. Tasks include verifying that you have no pending workflows, stopping the Siebel Server, and so on.	For an overview, see *Perform Preupgrade Tasks for the Siebel Database*. For details on basic database preparation, see *Preparing for Siebel Database Upgrade*.	Yes	Yes
7. Preparing Siebel Business Applications data for an upgrade.	See *Preparing Siebel Application Data for Upgrade*.	Yes	Yes
8. Prepare your developers for the upgrade. Tasks include backing up development databases, ensuring that all developers' projects are checked in and unlocked.	See *Prepare Developers for the Upgrade*.	Yes	No
9. Use the previous version of Siebel Tools to rename your Siebel Repository.	For an overview, see *Upgrade Siebel Database Schema (upgrep)*. For task details, see *Renaming the Siebel Repository*.	Yes	No
10. Upgrade the Siebel database schema. Tasks include running the Database Configuration Wizard (upgrep) and the Upgrade Wizard.	For an overview, see *Upgrade Siebel Database Schema (upgrep)*.	Yes	Yes

Upgrade Tasks	Where to Find Documentation	Dev?	Prod
	For specific task details, including running the Database Configuration Utilities, see *Upgrading the Siebel Database*.		
11. Prepare for the repository merge.	For details, see *Perform Repository Merge*.	Yes	No
12. Use the newly installed version of Siebel Tools to perform the repository merge.	For details, see *Perform Repository Merge*.	Yes	No
13. Upgrade custom database schema. Tasks include running the Database Configuration Wizard to upgrade previous database schema customizations, running the Upgrade Wizard, resolving errors with upgrade log files, and backing up the Siebel database.	For an overview, see *Upgrade Custom Database Schema (upgphys)*. For task details, see *Upgrading the Siebel Database*.	Yes	No
14. Review the user interface. Tasks include reviewing potential object property conflicts, noting inherited behavior, and performing other tasks on the Siebel user interface.	For an overview, see *Review the User Interface*. For task details, see *Reviewing the Siebel User Interface*.	Yes	Yes
15. Create or update the Siebel Runtime Repository to assist in user interface testing.	For details, see *Using Siebel Tools* .	Yes	Yes
16. Perform postmerge development tasks.	For an overview, see *Perform Postmerge Development Tasks*. For task details, see *Siebel Postmerge Development Tasks*.	Yes	No
17. Perform postupgrade tasks on database and file system. These tasks are done following a completed upgrade.	For an overview, see *Perform Postupgrade Tasks for Database and File System*. For task details, see *Postupgrade Tasks for the Siebel Database*.	Yes	Yes
18. Perform postupgrade tasks for applications configuration. Tasks include installing and deploying any workflows, as well as performing other postupgrade configuration tasks.	For an overview, see *Perform Postupgrade Tasks for Applications Configuration*. For task details, see *Postupgrade Tasks for Siebel Business Applications*.	Yes	Yes

Testing the Environment

In this phase, perform system tests and prepare for the transition to the next database environment. The following table includes information relevant to this phase of the upgrade.

Upgrade Tasks	Where to Find Documentation	Dev?	Prod?
1. Perform system tests. Use available test data to perform unit testing.	For an overview, see *Perform System Tests*.	Yes	Yes
2. Prepare for the transition to the next environment. If you upgraded a development environment, then you will move to the production test environment, if you upgraded the production test environment then you proceed to the production environment. Tasks include creating or updating a Siebel Runtime Repository, and regenerating Siebel repository definition files, if necessary.	For an overview, see *Prepare for Transition to Production Test Environment*. For details on creating or updating a Siebel Runtime Repository, see *Using Siebel Tools* . For details on regenerating Siebel repository definition files, see *Regenerating the Siebel Repository Definition Files*.	Yes	Yes

Implementing the Environment

Implementing your environment occurs when you have fully upgraded and tested your production test environment and then move to your live, production environment. The following table includes information relevant to this phase of the upgrade.

Note: Perform these tasks only if you are transitioning from a production test to a production environment.

Upgrade Tasks	Where to Find Documentation	Dev?	Prod?
1. Apply the latest maintenance release (if any).	Check with your Oracle representative for any information on Siebel CRM maintenance releases which might be applicable to your release.	No	Yes
2. Perform an architecture and infrastructure review to ensure the production environment meets the requirements of the upgrade.	For help with an architecture and infrastructure review, contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.	No	Yes

Process of Upgrading a Siebel Development Environment

Platforms: Windows and UNIX only. This topic does not apply to IBM z/OS.

This topic lists the steps required to upgrade a Siebel development environment to the current release. Print this topic and use it as a checklist when doing the upgrade.

Note: Links to other topics in this guide are provided throughout this chapter. These links are meant to provide additional information that cannot be provided in these brief overviews. You must, however, perform the upgrade in the proper order. For information on general planning, see this chapter as well as *Siebel Database Upgrade Planning*, and *Application Planning for a Siebel Upgrade* To begin the process of upgrading your Siebel database, see *Preparing for Siebel Database Upgrade*, then refer to each subsequent chapter.

The topic is divided into sections, each containing a list of numbered steps. You complete each section in the order shown.

Note: If you are performing a development environment upgrade on IBM z/OS, then see *Siebel Database Upgrade Guide for DB2 for z/OS* .

Search for Articles on My Oracle Support

- Check My Oracle Support for recently published articles regarding the upgrade. For more information, see *Naming Conventions Used in This Guide*.

Upgrade the Servers

Verify that you have identified all of the Siebel releases that are required for the upgrade. Along with the current release, also install the latest Siebel CRM monthly update. For more information on server upgrade and installation, see *Siebel Installation Guide* . See also the Certifications tab on My Oracle Support.

CAUTION: Do not install a new Siebel database as part of upgrading the Siebel Enterprise.

To perform the following steps, see *Siebel Installation Guide* .

1. Upgrade the Siebel Gateway, Siebel Servers, and Siebel Application Interface.

   For information on installing these Siebel Enterprise components, see *Siebel Installation Guide* . The modules named are for the current release.
2. Choose the option to install the Siebel Database Configuration Utilities along with the Siebel Gateway and Siebel Server.
3. Install language packs for your currently deployed languages and any new languages.
4. If you have customized the configuration of Enterprise components, such as Siebel Servers, then you can run a script to migrate configuration parameters to the upgraded Siebel Enterprise. (Do not perform this step for a migration installation, which is described in *Siebel Installation Guide* .)
5. If you are upgrading to Siebel CRM versions 8.1 or 8.2, then copy the ancestor repositories. In Siebel CRM versions 8.1 and 8.2, the ancestor repository files are located in an Ancestor Repositories media directory that you access from Oracle Software Delivery Cloud. For more information, see *Preparing for the Repository Merge*.

Upgrade Third-Party Software

- Upgrade third-party software as required due to dependencies on Oracle's Siebel software or other installed software. For example, you might need to upgrade the following software:
    - Operating system software. Some database upgrades require newer versions of operating systems, such as AIX or Windows.

Upgrade the RDBMS

- If required, then upgrade the RDBMS version. Refer to the vendor's documentation to perform the upgrade:
    - For information on supported RDBMS systems, see the Certifications tab on My Oracle Support.
    - For information on how upgrading your RDBMS affects the upgrade process, see *About Upgrading Your RDBMS in the Siebel Environment*.

Perform Preupgrade Tasks for the Siebel Database

These steps apply to all database types.

1. Review guidelines for configuring the RDBMS. See the *Siebel Installation Guide* .
2. Verify that the Workflow Monitor and Workflow action agents have processed all pending requests.
3. Stop Siebel Servers, but make sure the Siebel Gateway is running.

    Note: The Siebel Gateway must be running for the Database Configuration Wizard to proceed.

4. Perform the tasks in *Preparing for Siebel Database Upgrade*

Perform Preupgrade Tasks for IBM DB2

1. Perform the tasks in *Preparing an IBM DB2 Database for a Siebel Upgrade*
2. If you are disabling archive logging, then allow time for a cold backup when reenabling archive logging.
3. Apply runstats to all tables, reorganize any that are out of cluster, or that are clustered less than 90%, then execute runstats on those tables again. This will improve performance and reduce the number of tasks that will need to be done after the upgrade is complete.

Perform Preupgrade Tasks for Oracle Database

1. Perform the tasks in *Preparing an Oracle Database for a Siebel Upgrade*
2. Run the Oracle Analyze command on the Siebel database. Highly fragmented indexes can cause the upgrade to fail.

Perform Preupgrade Tasks for Microsoft SQL Server

1. Perform the tasks in *Preparing a Microsoft SQL Server Database for a Siebel Upgrade*
2. Run Microsoft SQL Server statistics. This will improve upgrade performance.

Perform Preupgrade Tasks for Application Data

Perform the tasks in *Preparing Siebel Application Data for Upgrade*

Prepare Developers for the Upgrade

To prepare developers for the upgrade, do the following.

1. Back up the development database.
2. Verify that all developers have checked in their projects and that all projects are unlocked.
3. Verify that all developers have disconnected from the database. The only open account must be the account that is performing the upgrade.
4. Install the new version of Siebel Tools on development workstations.

   Note: Keep at least one copy of the previous version of Siebel Tools. You will need it to perform repository operations before the repository merge.

Upgrade Siebel Database Schema (upgrep)

1. (Optional) *Changing the Siebel Database Configuration Utilities Language*.
2. Run the Database Configuration Wizard:
   - *Preparing to Run the Siebel Database Configuration Wizard*.
   - *Running the Siebel Database Configuration Wizard on Windows*.
   - *Running the Siebel Database Configuration Wizard on UNIX*.

     Choose the following settings:
   - Environment Type**:** Development
   - Upgrade Options**:** Upgrade Siebel Database Schema (upgrep)
3. Edit generated SQL files as required by articles on My Oracle Support, and by the applicable version of *Siebel Release Notes* on My Oracle Support.
4. Resume the upgrade: *Starting the Siebel Upgrade Wizard*.
5. Review the upgrade logs and resolve errors:
   - *Reviewing Siebel Upgrade Log Files for Errors*.
6. If the upgrade contains unacceptable errors, then do the following:
   a. Restore the backup of the database.

   b. Correct the errors.
   c. If errors occurred because you entered incorrect information in the Database Configuration Wizard, then see *Regenerating SQL Files for a Siebel Upgrade*.
   d. Rerun the Database Configuration Wizard.
7. Back up the upgraded database.

Prepare for Repository Merge

1. *Preparing for the Repository Merge*.

   > **Note:** If you are upgrading from Siebel CRM version 8.1.1.x (SIA repository), version 8.2.2.x, version 15.x, or version 16.x to version 17.0, then use *Process of Meeting Requirements for an Incremental Repository Merge*.

2. *Migrating Siebel Repository Objects to the Standard User Interface*.
3. Set the Upgrade Ancestor property for copied objects. See *Configuring Siebel Repository Objects to Inherit Upgrade Behavior*.

Perform Repository Merge

1. *Performing a Siebel Repository Merge* .

   > **Note:** If you are upgrading from Siebel CRM version 8.1.1.x (SIA repository), version 8.2.2.x, version 15.x, or version 16.x to version 17.0, then see for example, *Executing Incremental Repository Merge on UNIX*.

2. *Reviewing the Siebel Repository Merge Log Files*.
3. If the repository merge contains unacceptable errors, then do the following:
   a. Restore the backup of the upgraded database.
   b. Correct the errors.
   c. Rerun the repository merge.

   > **Note:** Before completing the next two steps (Step 4 and Step 5), you must execute the Full Publish command in Siebel Tools to generate the Runtime Repository data.

4. *Running the Siebel Postmerge Utilities*.
   This step is required only for full repository merge and not if you are performing an incremental repository merge.
5. *Generating Siebel Enterprise Integration Manager Temporary Columns*.
6. (Optional) Configure case insensitive queries in the New Customer Repository. See *Running the Siebel Case Insensitivity Wizard*.
7. Back up the Siebel database.

Upgrade Custom Database Schema (upgphys)

1. Run the Database Configuration Wizard:
   - *Preparing to Run the Siebel Database Configuration Wizard*.
   - *Running the Siebel Database Configuration Wizard on Windows*.

- *Running the Siebel Database Configuration Wizard on UNIX*.

  Choose the following settings:
- **Upgrade Options:** Upgrade Custom Database Schema (upgphys)
- **Environment Type:** Development

2. Run the Upgrade Wizard: *Starting the Siebel Upgrade Wizard*.
3. Review the upgrade log files and resolve errors: *Reviewing Siebel Upgrade Log Files for Errors*.
4. If the upgrade contains unacceptable errors, then do the following:
   a. Restore the backup of the database you made after the repository merge.
   b. Correct the errors.
   c. Repeat the steps beginning after the backup.
5. *Manually Archiving Siebel Upgrade Log Files*.
6. Back up the upgraded database.

Review the User Interface

1. *Reviewing Siebel Repository Object Property Conflicts*.
2. Review the following topics on user interface upgrade before proceeding:
   - *About Inheriting Upgrade Behavior in a Siebel Upgrade*
   - *About the Siebel Postmerge Utilities*
3. Create or update the Siebel Runtime Repository to assist in user interface testing. For more information, see *Using Siebel Tools* .
4. If you customized style sheet or web template files in the previous release, implement those customizations in the new release again and execute a Full Publish in Siebel Tools to compile the Siebel Runtime Repository data.

   Carefully review the user interface in the new release before reimplementing customizations to these files.
5. Perform the tasks in *Reviewing the Siebel User Interface*
6. The postmerge utilities do not convert certain types of flow-based applets to grid-based applets. For example, they do not convert custom form applets to grid-based applets. See the editing applet layout topic in *Configuring Siebel Business Applications* and convert the remaining flow-based applets as required.

Perform Postmerge Development Tasks

1. Perform the tasks in *Siebel Postmerge Development Tasks*
2. Resolve any business component and join conflicts.
3. *Deleting Unneeded Siebel Repository Files*.

Perform Postupgrade Tasks for Database and File System

1. Perform the tasks in *Postupgrade Tasks for the Siebel Database*
2. Reset upgrade-specific database and database server parameters back to their recommended settings for production. See *Siebel Installation Guide* for recommended parameter settings.
3. If you exported data from interface tables before the upgrade, then review the database and import the data as desired. See *Siebel Enterprise Integration Manager Administration Guide* .

4. Generate a Siebel Remote database template file. See *Siebel Remote and Replication Manager Administration Guide* .
5. Run database statistics.

Note: The upgrade is complete. The remaining topics deal with configuration and validation tasks.

Perform Postupgrade Tasks for Applications Configuration

1. If applicable, then review the results of the Person and Organization merge. Make configuration changes as required.
2. Perform the tasks in *Postupgrade Tasks for Siebel Business Applications*
3. Verify the function of interfaces in integrated applications.
4. Deploy workflows.
5. If you have set up integrations for transferring data to or from third-party applications using Siebel Enterprise Application Integration, then verify the integrations are configured correctly. For information on EAI, see *Overview: Siebel Enterprise Application Integration* .
6. If you have used Siebel Enterprise Integration Manager to set up batch processing jobs, then verify Siebel Enterprise Integration Manager is configured correctly. For information on Siebel Enterprise Integration Manager, see *Siebel Enterprise Integration Manager Administration Guide* .

Perform System Tests

- Use available test data to perform unit testing. Validate the operation of the application in the following areas:
  - User interface
  - Data interfaces
  - Integrity of migrated data
  - Workflow function

Prepare for Transition to Production Test Environment

1. Publish the Siebel Runtime Repository. For more information, see *Using Siebel Tools* .
2. If you revised the repository after running upgphys, then you must regenerate the repository definition files. See *Regenerating the Siebel Repository Definition Files*.

Process of Upgrading a Siebel Production Test Environment

Platforms: Windows and UNIX only. This topic does not apply to IBM z/OS.

This topic lists the steps required to upgrade a production test environment to the current release. Print this topic and use it as a checklist when doing the upgrade.

Note: Links to other topics in this guide are provided throughout this chapter. These links are meant to provide additional information that cannot be provided in these brief overviews. You must, however, perform the upgrade in the proper order. For information on general planning, see this chapter as well as *Siebel Database Upgrade Planning* and *Application Planning for a Siebel Upgrade* To begin the process of upgrading your Siebel database, see *Preparing for Siebel Database Upgrade* then refer to each subsequent chapter after that.

The topic is divided into sections, each containing a list of numbered steps. Complete each section in the order shown.

Note: If you are performing a Production Test environment upgrade on IBM z/OS, then see *Siebel Database Upgrade Guide for DB2 for z/OS* .

Search for Articles on My Oracle Support

- Check My Oracle Support for recently published articles regarding the upgrade. For more information, see *Naming Conventions Used in This Guide*.

Upgrade the Servers

Verify that you have identified all of the Siebel releases that are required for the upgrade. Along with the current release, also install the latest Siebel CRM monthly update. For more information on server upgrade and installation, see *Siebel Installation Guide* . See also the Certifications tab on My Oracle Support.

CAUTION: Do not install a new Siebel database as part of upgrading the Siebel Enterprise.

To perform the following steps, see *Siebel Installation Guide* :

1. Upgrade the Siebel Gateway, Siebel Servers, and Siebel Application Interface.

   For information on upgrading these Siebel Enterprise components, see *Siebel Installation Guide* . The modules named are for the current release.
2. Install the Siebel Database Server files on the Siebel Server you will use to perform the upgrade. You only need to install the database server files for the database type that you are upgrading.

   Note: You need only to install a new Siebel Server instance if you are using a different computer from that on which you performed the Production Test Upgrade. If you are using the same Siebel Server installation as for the Production Test upgrade, then the existing files can be leveraged instead.
3. Install language packs for your currently deployed languages and any new languages.
4. If you have customized the configuration of enterprise components, such as Siebel Servers, then you can run a script to migrate configuration parameters to the upgraded Siebel Enterprise. (Do not perform this step for a migration installation, which is described in *Siebel Installation Guide* .)

Upgrade Third-Party Software

- Upgrade third-party software as required due to dependencies on Oracle's Siebel software or other installed software. For example, you might need to upgrade the following software:

Operating system software. Some database upgrades require newer versions of AIX or Windows.

Upgrade the RDBMS

- If required, upgrade the RDBMS version. Refer to the vendor's documentation to perform the upgrade:
  - For information on supported RDBMS systems, see the Certifications tab on My Oracle Support.
  - For information on how upgrading your RDBMS affects the upgrade process, see *About Upgrading Your RDBMS in the Siebel Environment*.

Perform Preupgrade Tasks for the Siebel Database

These steps apply to all database types.

1. Review guidelines for configuring the RDBMS. See the *Siebel Installation Guide* .
2. Verify that the Workflow Monitor and Workflow action agents have processed all pending requests.
3. Perform the tasks in *Preparing for Siebel Database Upgrade*

Perform Preupgrade Tasks for IBM DB2

1. Perform the tasks in *Preparing an IBM DB2 Database for a Siebel Upgrade*
2. Make sure that runstats is current in your environment and that tables with clustering indexes are not out of cluster. Reorganize any tables that are out of cluster or that are clustered less than 90% before the upgrade if possible.

   Table maintenance must be aggressively pursued prior to the upgrade to ensure that your system is in optimal condition. The same general consideration must also be kept in mind with regard to repository tables.

Perform Preupgrade Tasks for Oracle Databases

1. Perform the tasks in *Preparing an Oracle Database for a Siebel Upgrade*
2. Run the Oracle Analyze command on the Siebel database. Highly fragmented indexes can cause the upgrade to fail.

Perform Preupgrade Tasks for Microsoft SQL Server

1. Perform the tasks in *Preparing a Microsoft SQL Server Database for a Siebel Upgrade*
2. Run Microsoft SQL Server statistics. This will improve upgrade performance.

Perform Preupgrade Tasks for Application Data

Perform the tasks in *Preparing Siebel Application Data for Upgrade*

Prepare the Siebel Database for Upgrade

1. Stop the Siebel Servers and the Siebel Gateway.
2. Close all database connections. The only database connection must be the account performing the upgrade.
3. Copy application files to the environment, including:
   - Reports files.
4. Copy the custrep.dat and schema.ddl files to the environment. See *Moving the Siebel Repository Files*.

   If you have made changes to the Siebel repository since performing the development upgphys, then you must regenerate the schema.ddl and custrep.dat files. See *Regenerating the Siebel Repository Definition Files*.
5. Verify the production test database is either a copy of the current production database or has the same topology and amount of data.

   This is important for effective upgrade tuning before performing the production upgrade.
6. Back up the production test environment database. (If you backed up the database as part of an RDBMS upgrade, then ignore this step.)

   To do upgrade tuning, you will restore this database and perform test-upgrades on it.

Upgrade the Siebel Database (upgrep and upgphys)

1. Run the Database Configuration Wizard:
   - *Preparing to Run the Siebel Database Configuration Wizard*.
   - *Running the Siebel Database Configuration Wizard on Windows*.
   - *Running the Siebel Database Configuration Wizard on UNIX*.

   Choose the following settings:
   - **Environment Type:** Production
   - **Upgrade Options:** Upgrade Siebel Database Schema (upgrep and upgphys)

   See *Performing a Production Test or Production Environment Migration from Siebel CRM 8.1.1.x (SIA Repository)*.
2. Resume the upgrade. See *Starting the Siebel Upgrade Wizard*.
3. Review the upgrade logs and resolve errors. See *Reviewing Siebel Upgrade Log Files for Errors*.
4. If the upgrade contains unacceptable errors, then do the following:
   a. Restore the backup of the database.
   b. Correct the errors.
   c. If errors occurred because you entered incorrect information in the Database Configuration Wizard, then see *Regenerating SQL Files for a Siebel Upgrade*.
   d. Rerun the Database Configuration Wizard.
5. See *Manually Archiving Siebel Upgrade Log Files*.
6. Back up the upgraded database.

Refer to the following supplemental documentation for any extra instructions or considerations for your specific upgrade scenario:

- Current Monthly Update Guide and Release Notes. *Siebel Database Upgrade Guide*
- My Oracle Support

Perform Postupgrade Tasks for Database and File System

1. Perform the tasks in *Postupgrade Tasks for the Siebel Database*
2. Reset upgrade-specific database and database server parameters back to their recommended settings for production. See *Siebel Installation Guide* for recommended parameter settings.
3. If you exported data from interface tables before the upgrade, then review the database and import the data as desired.
4. Generate a Siebel Remote database template file. See *Siebel Remote and Replication Manager Administration Guide* .
5. Run database statistics.

Note: The upgrade is complete. The remaining topics deal with configuration and validation tasks.

Perform Postupgrade Tasks for Applications Configuration

1. Perform the tasks in *Postupgrade Tasks for Siebel Business Applications*
2. Verify the function of interfaces in integrated applications.
3. Deploy workflows.
4. If you have set up integrations for transferring data to or from third-party applications using Siebel Enterprise Application Integration, then verify that the integrations are configured correctly. For information on EAI, see *Overview: Siebel Enterprise Application Integration* .
5. If you have used Siebel Enterprise Integration Manager to set up batch processing jobs, then verify Siebel Enterprise Integration Manager is configured correctly. For information on Siebel Enterprise Integration Manager, see *Siebel Enterprise Integration Manager Administration Guide* .
6. If you customized style sheet or Web template files in the previous release, then reimplement those customizations in the new release as desired.

   Carefully review the user interface in the new release before reimplementing customizations to these files.

Perform System Tests

- Use available test data to perform unit testing. Validate the operation of the application in the following areas:
  - User interface
  - Data interfaces
  - Integrity of migrated data
  - Workflow function

Process of Tuning Siebel Upgrade Performance

Platforms: Windows and UNIX only.

Note: This process is optional.

Use this process to run test upgrades in the production test environment so you can tune upgrade performance. Improving upgrade performance reduces downtime when you perform the production environment upgrade. The steps in this process cover standard performance tuning. To implement more advanced tuning, including high-availability tuning, contact your Oracle sales representative for Oracle Advanced Customer Services.

Also use this process to test the additive schema changes feature. This feature allows you to perform part of the upgrade on the production database without taking it offline. This reduces the downtime required to upgrade the production database. Test the additive schema changes feature to verify that it does not adversely affect the operation of the application.

Perform this process in the production test environment. Do not perform this process in the production environment.

Note: Links to other topics in this guide are provided throughout this chapter. These links are meant to provide additional information that cannot be provided in these brief overviews. You must, however, perform the upgrade in the proper order. For information on general planning, see this chapter as well as *Siebel Database Upgrade Planning* and *Application Planning for a Siebel Upgrade* To begin the process of upgrading your Siebel database, see *Preparing for Siebel Database Upgrade* then refer to each subsequent chapter.

Review the following upgrade planning and performance tuning resources before performing this process:

- *About Siebel Additive Schema Changes Mode*
- **478308.1 (Article ID) on My Oracle Support.** This document was previously published as Siebel Technical Note 616. This article describes the strategies for minimizing production environment downtime during an upgrade. The steps that follow are intended primarily for use with the baseline guidelines described in 478308.1 (Article ID).
- *Tuning the Siebel Upgrade Files* This chapter provides detailed information on how to use Upgrade Tuner.

UNIX deployments. You must have a Siebel Server installed on a Windows host to run the Upgrade Tuner. To obtain a Windows version of Siebel Server, contact your account manager or Oracle Global Customer Support.

CAUTION: Do not use this topic to tune upgrade performance on IBM z/OS. Instead, refer to *Siebel Database Upgrade Guide for DB2 for z/OS* .

Set Up the Target Database

1. Back up and remove the upgraded production test database.

   This preserves seed data and metadata changes you have made that must be migrated to the production environment. This database is called the production-test final database.

2. In the production test environment, install a recent backup of your production database.

   This database has not been upgraded and is called the target database. You will use it to perform test upgrades as part of tuning upgrade performance.
3. Define an ODBC connection to the target database.
4. Verify that the target database and RDBMS server are configured for optimum upgrade performance:

   - *Preparing an IBM DB2 Database for a Siebel Upgrade*
   - *Preparing an Oracle Database for a Siebel Upgrade*
   - *Preparing a Microsoft SQL Server Database for a Siebel Upgrade*
5. *Preparing Siebel Tables and Views for Upgrade*.
6. Run statistics on the target database. This optimizes query performance.
7. **UNIX.** *Securing AIX Memory Allocation Segment Space for the Siebel Database*.
8. Perform the tasks in *Preparing Siebel Application Data for Upgrade*

Test Additive Schema Changes

If you do not plan to run additive schema changes on the production database in the production environment, then you do not need to perform the steps in this topic.

Perform these steps on the target database.

1. Review *About Siebel Additive Schema Changes Mode*.
2. (Optional) *Changing the Siebel Database Configuration Utilities Language*.
3. In the production test environment, run the Database Configuration Wizard:

   - *Preparing to Run the Siebel Database Configuration Wizard*
   - *Running the Siebel Database Configuration Wizard on Windows*
   - *Running the Siebel Database Configuration Wizard on UNIX*
4. Choose Apply Additive Schema Changes mode.

   The Apply Additive Schema Changes mode identifies changes that can be made to the target database without taking it offline.

   Windows. You will not be prompted whether you want to run the Upgrade Wizard. Instead, the Upgrade Wizard starts automatically, and creates the `schema.additive.sql` script. The Upgrade Wizard does not run the script against the target database.
5. **UNIX.** Run the Siebel Upgrade Wizard.

   See *Starting the Siebel Upgrade Wizard*.

   Specify `master_additive_gen.ucf` as the input file.

   The Upgrade Wizard generates the `schema.additive.sql` script. It does not run the script against the target database.
6. Run the SQL script against the target database.

   See *Applying Siebel Additive Schema Changes*.
7. Thoroughly review the operation of all applications. Verify that applying the additive schema changes has not adversely affected the operation of all applications.

8. If applying additive schema changes adversely affects the operation of the applications, then do the following:
    a. For help diagnosing the cause of the problem and help in taking corrective action create a service request (SR) on My Oracle Support. Alternatively, you can phone Global Customer Support directly to create a service request or get a status update on your current SR. Support phone numbers remain the same and are listed on My Oracle Support.
    b. Restore the target database from backup before proceeding with further upgrade tuning.
    c. Do not use the additive schema changes feature during your production upgrade.
9. If applications function normally, then continue with upgrade tuning. Consider using the additive schema changes feature during your production upgrade.

Upgrade the Target Database (upgrep and upgphys)

1. (Optional) *Changing the Siebel Database Configuration Utilities Language*.
2. In the production test environment, run the Database Configuration Wizard:
    - *Preparing to Run the Siebel Database Configuration Wizard*
    - *Running the Siebel Database Configuration Wizard on Windows*
    - *Running the Siebel Database Configuration Wizard on UNIX*

    Choose the following settings:
    - **Upgrade Options:** Upgrade Siebel Database Schema (upgrep and upgphys)
    - **Environment Type:** Production
3. In the production test environment, run Siebel Upgrade Wizard.

    See *Starting the Siebel Upgrade Wizard*.

    Note the time required to upgrade the database.
4. Review the upgrade log files for errors: *Reviewing Siebel Upgrade Log Files for Errors*.
5. If the upgrade contains errors that prevented completion or adversely affected performance, then correct the errors and rerun the upgrade.
6. *Manually Archiving Siebel Upgrade Log Files*.

Tune the Upgrade Files

1. Evaluate upgrade performance, particularly the time required to complete the upgrade.

    During a production mode upgrade, using the /z and /h parameters in the import command will reduce upgrade time and improve the overall repository import process performance.
2. Do one of the following:
    - If the time required to complete the upgrade is acceptable, then no further tuning is needed. Perform the steps in *Restore the Production-Test Final Database*.
    - If the time required to complete the upgrade is too long, then perform the remaining steps in this topic to continue tuning upgrade performance.
    - If the time required to complete the upgrade is too long and you cannot tune further, then contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services to apply advanced tuning.

3. Carefully review target database and RDBMS server configuration. Adjust as needed to further improve upgrade performance.
4. Run Upgrade Tuner to tune the upgrade files:

   - **UNIX.** *Transferring UNIX Files for Use by Siebel Upgrade Tuner*.
   - *Starting and Stopping Siebel Upgrade Tuner*.
   - *Managing Parallel Threads Using Siebel Upgrade Tuner*.
   - *Managing Zero-Row SQL Commands Using Siebel Upgrade Tuner*.

   To roll back changes you have made in previous Upgrade Tuner sessions, see *Rolling Back Siebel Upgrade Tuner Changes*.

Restore the Target Database

Perform these steps if you have made changes to the upgrade environment or to the upgrade files and want to run the upgrade again to verify performance improvement.

1. In the production test environment, restore the target database from backup.

   This returns the target database to its non-upgraded state so that you can perform another test upgrade.
2. In the production test environment, apply additive schema changes again as desired.
3. Perform another test upgrade and evaluate upgrade performance.
4. Repeat the tuning process and perform test-upgrades until upgrade performance is acceptable.

Restore the Production-Test Final Database

Perform these steps only if you have completed tuning upgrade performance.

1. In the production test environment, delete and remove the target database.
2. In the production test environment, restore the production-test final database from backup.

Process of Upgrading a Siebel Production Environment

Platforms: Windows and UNIX only. This topic does not apply to IBM z/OS.

To perform this process, you must be able to execute ODBC commands on the production database from within the production environment. For more information, see *About the Siebel Database Configuration Wizard Utilities*. For help executing ODBC commands within the production test environment, create a service request (SR) on My Oracle Support.

> **Note:** Links to other topics in this guide are provided throughout this chapter. These links are meant to provide additional information that cannot be provided in these brief overviews. You must, however, perform the upgrade in the proper order. For information on general planning, see this chapter as well as *Siebel Database Upgrade Planning* and *Application Planning for a Siebel Upgrade* To begin the process of upgrading your Siebel database, see *Preparing for Siebel Database Upgrade* then refer to each subsequent chapter.

This topic lists the tasks required to transition your production test environment to production. Print this topic and use it as a checklist when doing the upgrade. Complete the steps in the order shown.

> **Note:** If you are performing a production environment upgrade on IBM z/OS, then see *Siebel Database Upgrade Guide for DB2 for z/OS* .

Search for Articles on My Oracle Support

- Check My Oracle Support for recently published articles regarding the upgrade. For more information, see *Naming Conventions Used in This Guide*.

Upgrade the Servers

Verify that you have identified all of the Siebel CRM releases that are required for the upgrade. Along with the current release, also install the latest Siebel CRM monthly update. For more information on server upgrade and installation, see *Siebel Installation Guide* . See also the Certifications tab on My Oracle Support.

> **CAUTION:** Do not install a new Siebel database as part of upgrading the Siebel Enterprise.

To perform the following steps, see the *Siebel Installation Guide* .

1. Upgrade the Siebel Gateway, Siebel Servers, and Siebel Application Interface.

   For information on upgrading these Siebel Enterprise components, see *Siebel Installation Guide* . The modules named are for the current release.
2. Install the Siebel Database Server files on the Siebel Server you will use to perform the upgrade. You only need to install the database server files for the database type that you are upgrading.
3. Install the language packs for your currently deployed languages and any new languages.
4. If you have customized the configuration of enterprise components, such as Siebel Servers, then you can run a script to migrate configuration parameters to the upgraded Siebel Enterprise. (Do not perform this step for a migration installation, which is described in the *Siebel Installation Guide* .)

Upgrade Third-Party Software

- Upgrade third-party software as required due to dependencies on Oracle's Siebel software or other installed software. For example, you might need to upgrade the following software:
  - Operating system software. Some database upgrades require newer versions of AIX or Windows.

Apply Additive Schema Changes

These steps are optional. Perform these steps only after you have thoroughly tested the additive schema changes feature in the production test environment and have verified that it will not adversely affect your applications.

Perform these steps on the production database. You do not need to take the production database offline. You can perform these steps anytime prior to running the production upgrep.

After applying additive schema changes there is no need to run database statistics. The schema changes do not affect the performance of your release prior to the upgrade.

1. Review *About Siebel Additive Schema Changes Mode*.
2. Verify that you have a recent backup of the production database and that you are logging transactions.

   If you encounter problems applying additive schema changes, then you might need to restore the production database.
3. (Optional) *Changing the Siebel Database Configuration Utilities Language*.
4. In the production environment, start the Database Configuration Wizard:
   - *Preparing to Run the Siebel Database Configuration Wizard*
   - *Running the Siebel Database Configuration Wizard on Windows*
   - *Running the Siebel Database Configuration Wizard on UNIX*
5. Choose Apply Additive Schema Changes mode.

   The Apply Additive Schema Changes mode identifies changes that can be made to the production database without taking it offline.

   Windows. You will not be prompted whether you want to run the Upgrade Wizard. Instead, the Upgrade Wizard starts automatically, and creates the `schema.additive.sql` script. The Upgrade Wizard does not run the script against the production database.
6. UNIX. Run the Siebel Upgrade Wizard.

   See *Starting the Siebel Upgrade Wizard*.

   Specify `master_additive_gen.ucf` as the input file.

   The Upgrade Wizard generates the `schema.additive.sql`script. It does not run the script against the production database.
7. Run the SQL script against the production database.

   See *Applying Siebel Additive Schema Changes*.
8. Thoroughly review the operation of all applications. Verify that applying additive schema changes has not adversely affected applications function.
9. If you are applying additive schema changes adversely affects the operation of the application, then do the following:
   a. Contact Oracle Global Support for help in diagnosing and correcting the problem.
   b. If required, restore the production database from backup.

Upgrade the RDBMS

- If required, upgrade the RDBMS version. Refer to the vendor's documentation to perform the upgrade:
  - For information on supported RDBMS systems, see the Certifications tab on My Oracle Support.
  - For information on how upgrading your RDBMS affects the upgrade process, see *About Upgrading Your RDBMS in the Siebel Environment*.

Perform Preupgrade Tasks for the Siebel Database

These steps apply to all database types.

1. Review guidelines for configuring the RDBMS. See the *Siebel Installation Guide* .
2. Verify that the Workflow Monitor and Workflow action agents have processed all pending requests.
3. Stop the Siebel Servers.
4. Verify that the Siebel Gateway is still running.
5. Stop all other Siebel Servers.
6. Close all database connections. The only open connection must be the account performing the upgrade.
7. Perform the tasks in *Preparing for Siebel Database Upgrade*

Perform Preupgrade Tasks for IBM DB2

1. Perform the tasks in *Preparing an IBM DB2 Database for a Siebel Upgrade*
2. Make sure that all transactional tables with clustering indexes are at least 90% clustered and stats are current.

   Note: You might be required to perform runstats on a small number of transactional tables and indexes following a successful upgrade. All Siebel Enterprise Integration Manager tables will have been replaced and might need statistics gathered.

Perform Preupgrade Tasks for Oracle Database

1. Perform the tasks in *Preparing an Oracle Database for a Siebel Upgrade*
2. Run the Oracle Analyze command on the Siebel database. Highly fragmented indexes can cause the upgrade to fail.

Perform Preupgrade Tasks for Microsoft SQL Server

1. Perform the tasks in *Preparing a Microsoft SQL Server Database for a Siebel Upgrade*
2. Run Microsoft SQL Server statistics. This will improve upgrade performance.

Perform Preupgrade Tasks for Application Data

Perform the tasks in *Preparing Siebel Application Data for Upgrade*

Perform Preupgrade Tasks for the User Interface

- Copy application files to the environment:
  - Reports files.

Upgrade the Siebel Database (upgrep and upgphys)

You do not run the upgrep and upgphys portions of the upgrade in the production environment. Instead, the numbered steps in this topic implement the following process:

- Run the Database Configuration Wizard in the production environment.
- In the utilities, enter the information for the production environment instead of the production test environment. For example, you enter the ODBC connection for the production environment.

  This information configures the driver file to run against the production database rather than the production test database. It also configures the driver file to use the upgrade SQL files you generated for the production test upgrade.
- Run the Upgrade Wizard. The Upgrade Wizard uses the SQL files in the production test environment to upgrade the database in the production environment.

  This approach has several advantages:
- You do not have to generate upgrade SQL files in the production environment, and then manually transfer the customizations to them from the production test environment.
- You do not lose any changes to the SQL files that were made by Siebel Upgrade Tuner in the production test environment.
- You do not have to run the Database Configuration Wizard in Prepare for Production mode again.
- With some exceptions, you do not have to perform again the database-related configuration tasks required by *Siebel Release Notes* on My Oracle Support or Articles on My Oracle Support.

If your network configuration prevents creating an ODBC connection to your production database from inside your production test environment, then contact Oracle Global Customer Support for assistance in completing the production upgrade.

1. Verify that you have a current backup of the production environment database.
2. On the Siebel Server you used to upgrade the production test environment, create an ODBC to connect to the production environment database.
3. Navigate to `DBSRVR_ROOT\common` (UNIX: `DBSRVR_ROOT/common`) and verify that the file sqlgen.usg exists.

   This file contains a record of when the SQL generator was run. When you run the Database Configuration Wizard, if this file exists, then no SQL commands are generated.

   CAUTION: If this file does not exist, then do not run the Database Configuration Wizard. The Database Configuration Wizard will overwrite the SQL files used to upgrade your production test database. Contact Oracle Global Customer Support for guidance on proceeding with the upgrade.

   You do not have to run the Database Configuration Wizard in Prepare for Production Mode.
4. (Optional) *Changing the Siebel Database Configuration Utilities Language*.
5. Run the Database Configuration Wizard:

   - *Preparing to Run the Siebel Database Configuration Wizard*
   - *Running the Siebel Database Configuration Wizard on Windows*
   - *Running the Siebel Database Configuration Wizard on UNIX*

   a. Choose the following settings when you run the utility:
      - **Upgrade Options.** Upgrade Siebel Database Schema (upgrep and upgphys)
      - **Environment Type.** Production
   b. Enter the information for the production environment instead of the production test environment.
   c. Enter the name of the ODBC for connecting to the production database.
   d. When prompted whether you want to run the Siebel Upgrade Wizard, answer No and exit.

      This updates the primary UCF file with the production environment configuration. When you run the Siebel Upgrade Wizard, the SQL commands will be executed on the production environment database.
6. In the production test environment, verify that the SQL scripts for performing the upgrade were not overwritten. You can do this by checking the modification times. If the scripts were overwritten, then do not continue. Instead, contact your Oracle sales representative for Oracle Advanced Customer Services to request assistance from Oracle's Application Expert Services.
7. Perform the production database upgrade. See *Starting the Siebel Upgrade Wizard*.

   The Siebel Upgrade Wizard uses the SQL commands generated for the production test environment to upgrade the production environment. If you used Upgrade Tuner to revise the SQL commands, then these changes are included.
8. Review the upgrade logs and resolve errors: *Reviewing Siebel Upgrade Log Files for Errors*.
9. If the upgrade contains unacceptable errors, then do the following:

   a. Restore the backup of the database.
   b. Correct the errors.
   c. Rerun the Siebel Upgrade Wizard.
10. *Manually Archiving Siebel Upgrade Log Files*.
11. Use the Siebel Application Deployment Manager to migrate administrative data such as LOVs and responsibilities from production test to production.
12. Back up the upgraded production environment database.

Perform Postupgrade Tasks for Database and File System

1. Perform the tasks in *Postupgrade Tasks for the Siebel Database*
2. Reset upgrade-specific database and database server parameters back to their recommended settings for production. See *Siebel Installation Guide* for recommended parameter settings.
3. If you exported data from interface tables before the upgrade, then review the database and import the data as desired.
4. Generate a Siebel Remote database template file. See *Siebel Remote and Replication Manager Administration Guide* .
5. Run database statistics.

Note: The upgrade is complete. The remaining topics deal with configuration and validation tasks.

Perform Postupgrade Tasks for Applications Configuration

1. If applicable, review the results of the Person and Organization merge. Make configuration changes as required.
2. Perform the tasks in *Postupgrade Tasks for Siebel Business Applications*

3. Verify the function of interfaces in integrated applications.
4. Deploy workflows.
5. If you have set up integrations for transferring data to or from third-party applications using Siebel Enterprise Application Integration, then verify that the integrations are configured correctly. For information on Siebel EAI, see *Overview: Siebel Enterprise Application Integration* .
6. If you have used Siebel Enterprise Integration Manager to set up batch processing jobs, then verify Siebel Enterprise Integration Manager is configured correctly. For information on Siebel Enterprise Integration Manager, see *Siebel Enterprise Integration Manager Administration Guide* .
7. If you customized the style sheet or Web template files in the previous release, then reimplement those customizations in the new release as desired.

Perform System Tests

- Use available test data to perform unit testing. Validate the operation of the application in the following areas:
  - User interface
  - Data interfaces
  - Integrity of migrated data
  - Workflow function

27 Siebel Case Insensitivity Wizard

Siebel Case Insensitivity Wizard

This chapter provides information about the Siebel Case Insensitivity Wizard. It includes the following topics:

- *Overview of What the Case Insensitivity Wizard Does*
- *How CIAI Columns and Indexes Are Implemented inthe Database*

Overview of What the Case Insensitivity Wizard Does

Environments: Development environment only.

Platforms: Windows, UNIX, IBM z/OS.

Databases: All databases.

The Case Insensitivity Wizard performs the following functions in the repository to configure columns to support CIAI queries. No columns or indexes are created in the Siebel database until you synchronize the repository to the Siebel database. The columns that you want to configure for CIAI queries are called base columns:

- Validates the syntax of all records if an input file is used.
- Validates that all specified tables and columns are eligible for CIAI configuration.
- For each eligible base column, defines a new CIAI column. The CIAI column contains the data in the base column converted to uppercase.
- If you select the Single or Copy All index strategy, then the wizard defines an index on the CIAI column.
- If you select the Copy All index strategy, the wizard defines a copy of all indexes that have the base column as a key. The new indexes reference the CIAI column instead of the base column.
- Sets the Default Insensitivity property for the base column to DB Case & Accent.
- Sets flags and performs other configuration operations in the repository to support CIAI queries.

The Case Insensitivity Wizard can also be run in a special mode to set the Default Insensitivity property on columns that do not have any indexes defined.

The main purpose of the CIAI query enhancement is to provide indexes that can be used for case insensitive searches. The database does not have to perform table scans to locate records. This enhancement allows the database to perform case insensitive searches more quickly.

For example, in S_CONTACT, you configure the column LAST_NAME for CIAI queries. The Case Insensitivity Wizard defines a column called LAST_NAME_CI. When you query for the name Smith, the Object Manager creates a query similar to the following (IBM DB2):

```
SELECT column_list FROM S_CONTACT
WHERE LAST_NAME_CI indicates SMITH
```

The database then uses the CIAI index on LAST_NAME_CI to locate the records.

For columns where Force Case is set, the data in the database is all in the same case. No CIAI columns or indexes are needed. The Object Manager uses the indexes already defined on the base column to retrieve records.

Related Topic

About the Siebel Case Insensitivity Wizard

Related Book

Configuring Siebel Business Applications

How CIAI Columns and Indexes Are Implemented in the Database

In a development environment upgrade, you apply the repository schema definition to the physical database. How CIAI columns and indexes in the repository are implemented in the physical database depends on the database type.

Oracle

On an Oracle database, CIAI columns are implemented as function-based indexes. For example, the following index would be created upon making S_CONTACT.FST_NAME a CIAI column:

```
create index S_CONTACT_M15_C1 on <tableowner>..S_CONTACT (NLS_UPPER("FST_NAME", 'NLS_SORT=GENERIC_BASELETTER'));
```

MSSQL

For Microsoft SQL Server databases, CIAI columns are implemented with the following two changes:

1. A new column with the suffix `_CI` is added to the table's definition for each column being enabled for CIAI queries. Note that MSSQL has a native capability to populate this column based on the source column.
2. A new index is created that leverages that column.

   For example, the following index would be created upon making S_CONTACT.FST_NAME a CIAI column:

   ```
   alter table dbo.S_CONTACT add FST_NAME_CI as FST_NAME COLLATE Latin1_General_CI_AI create nonclustered
   index S_CONTACT_M15_C1 on dbo.S_CONTACT ("FST_NAME")
   ```

DB2 (UDB and 390)

For DB2 databases, CIAI columns are implemented through the following steps:

1. A new column with the suffix `_CI` is added to the table's definition for each column being enabled for CIAI queries.
2. A one-time SQL statement is executed to populate this column for existing records.

   Note: If your implementation has Siebel Remote or Siebel Replication, you must re-extract all databases after running the CIAI Wizard.

3. An `insert` trigger prefixed with `CTI_` is created to populate the `_CI` column when new records are inserted.
4. An `update` trigger prefixed with `CTU_` is created to keep the `_CI` column up to date if a record is modified.

5. An index is created that leverages the `_CI` column.

DB2 UDB Example:

```
 alter table SIEBEL.S_CONTACT add "FST_NAME_CI" vargraphic(350);
update SIEBEL.S_CONTACT set FST_NAME_CI = UPPER(FST_NAME);
create trigger SIEBEL.CTI_S_CONTACT before insert on SIEBEL.S_CONTACT referencing new as n for each row
 begin atomic set n.FST_NAME_CI = UPPER(n.FST_NAME) end;
create trigger SIEBEL.CTU_S_CONTACT before update of FST_NAME on SIEBEL.S_CONTACT referencing new as n for
 each row
 begin atomic set n.FST_NAME_CI = UPPER(n.FST_NAME); end
create index SIEBEL.S_CONTACT_M15_C1 on SIEBEL.S_CONTACT ("FST_NAME_CI") PCTFREE 30 ALLOW REVERSE SCANS;
```

Note: The syntax for DB2/390 is very similar but includes additional storage information relevant only to the DBA.

Case Insensitivity Wizard Modes

You can run the Case Insensitivity Wizard in one of two modes:

- **Configure specified columns.** You can specify the columns you want to configure for CIAI queries by using an input file or by selecting the files manually:
    - **Use an input file.** The Case Insensitivity Wizard reads the input file and configures the columns in the file for CIAI queries. Using an input file allows you to control which configuration options the wizard uses. Oracle provides recommended input files. These files contain columns frequently used in searches. Use an input file if you want to enable large numbers of columns or to use methods or index strategies other than the defaults.
    - **Select columns manually.** You can use the Tools Object Explorer to manually select and configure specific columns for CIAI queries. The Case Insensitivity Wizard uses default settings to configure these columns. If you want to modify the configuration options, then you can export the configuration strings to a text file, edit them, and run the wizard using the edited file as an input file.
    - **Specify how queries are built for columns without indexes.** The Case Insensitivity Wizard defines CIAI columns and indexes only on columns that already have indexes defined. However, for columns without indexes that meet all other eligibility criteria, you can run the Case Insensitivity Wizard in a special mode to change the Default Insensitivity property from None to DB Case & Accent. In queries, the column values are then converted to uppercase before being compared. This allows searches to be both case and accent insensitive.

      For example, in S_CONTACT, assume the column LAST_NAME has no indexes defined on it. You run the Case Insensitivity Wizard to set Default Insensitivity to DB Case & Accent. When you query for the name Smith, the Object Manager uses a query similar to the following (IBM DB2):

      ```
      SELECT column_list FROM S_CONTACT WHERE UPPER(LAST_NAME) LIKE UPPER(Smith)
      ```

Input File Format

For configuring eligible table columns, the Case Insensitivity Wizard can accept a comma-delimited .csv file as input. Each line in the file is one record that defines one column to be configured for CIAI queries.

The record format is as follows:

```
Table_Name,Column_Name,Method,Index_Strategy,Operation
```

The fields in the record are explained in the following example of a record:

```
S_CONTACT,EMAIL_ADDR,Database,Copy All,On
```

Only the first two items in the record are mandatory (Table_Name and Column_Name). For records where items are omitted, the Case Insensitivity Wizard inserts the default value for the items.

If you omit an item from a record, then you must still provide the item's delimiting commas. Here is a record with the index strategy item omitted:

```
S_CONTACT,EMAIL_ADDR,Database,,On
```

The Case Insensitivity Wizard does not provide special handling for denormalized columns. To configure CIAI queries on denormalized columns, you must include them in an input file.

Oracle provides a recommended input file for Siebel Business Applications and for Siebel Industry Applications. The input files have a .csv extension and are located in the following directory:

Windows: `$SIEBEL_HOME\objects`

Eligibility Criteria

The Case Insensitivity Wizard verifies that all records in an input file or in columns that you manually select meet the following eligibility criteria:

- The table and column must exist in the repository.
- The column must be active and belong to the specified table.
- The table type must be supported for CIAI configuration.
- The column functional type must be supported for CIAI configuration.
- The column physical type must be supported for CIAI configuration.
- The column must have one or more indexes already defined on it. If no indexes are defined on the column, but the column is otherwise eligible, then the wizard accepts the column, but it does not create a CIAI column or any CIAI indexes for the column. The Case Insensitivity Wizard sets the Default Insensitivity to DB Case & Accent.

Table Name: Supported Table Types

The following table lists the table types that can be configured for CIAI queries.

Table Type	Can Be Configured for CIAI Queries?
Data (Intersection)	Yes
Data (Private)	Yes
Data (Public)	Yes

Table Type	Can Be Configured for CIAI Queries?
Database View	No
Dictionary	No
Extension	Yes
Extension (Siebel)	Yes
External	No
Interface	No
Log	No
Repository	No

Column Name: Supported Column Functional Types

The following table lists the column functional types that can be configured for CIAI queries.

Column Functional Type	Can Be Configured for CIAI Queries?
Data (Private)	Yes
Data (Public)	Yes
Denormalized	Yes
Extension	Yes
External	No
IFMGR: Exists	No
IFMGR: FKey	No
IFMGR: Status	No
IFMGR: ROW_ID	No

Column Functional Type	Can Be Configured for CIAI Queries?
System	No

Column Name: Supported Physical Column Types

The following table lists the physical column types that can be configured for CIAI queries.

Physical Type		
Database Value	**Maps To**	**Can Be Configured for CIAI Queries?**
C	Char	Yes
D	Date	No
N	Number	No
S	Time Stamp	No
T	Date Time	No
U	UTC Date Time	No
V	Varchar	Yes
X	Text	Yes
L	CLOB	Yes

Text and CLOB are accepted by the Case Insensitivity Wizard as valid. However, the wizard does not create a CIAI column or CIAI indexes for these two types. The wizard sets Default Insensitivity to DB Case & Accent.

Method and Index Strategy

The Case Insensitivity Wizard method and index strategy determine how the wizard configures CIAI queries for a column.

There are two methods:

- **Force Case.** The Force Case method does not create a CIAI column or indexes. Use the Force Case method for columns where the Force Case property is set.

  The Force Case property causes column data to be set to the same case (FirstUpper, lower, upper) before being written to the database. Since all the data in the base column are in the same case, the base column and its indexes can be used for case insensitive queries. A CIAI column and indexes are not needed.

  The Case Insensitivity Wizard considers Force Case to be set for a column when Force Case indicates FirstUpper, Lower, or Upper. The Case Insensitivity Wizard does not consider Force Case to be set when Force Case indicates none or is null.
- **Database.** The Database method defines a CIAI column for the base column and uses an *index strategy* to create indexes.

The index strategy in effect determines how indexes are defined for the CIAI column, as described in the following table.

Index Strategy	Description
None	The Case Insensitivity Wizard defines no new columns or indexes. The wizard sets the Default Insensitivity to DB Case & Accent.
Single	The Case Insensitivity Wizard defines a new CIAI column and defines a single index on it. For every index the base column participates in, the wizard does not create another index that references the CIAI column. The wizard sets the Default Insensitivity to DB Case & Accent.
Copy All	The Case Insensitivity Wizard defines a new CIAI column and a CIAI index for the column. In addition, for every index the base column participates in, the wizard defines a copy of that index. The copy references the CIAI column instead of the base column. The wizard sets the Default Insensitivity to DB Case & Accent.

For indexes with multiple columns as keys, for the first key where the column becomes CIAI enabled, the wizard defines a copy of the index. In the copy, the key references the CIAI column instead of base column. For each additional key that is CIAI enabled, the wizard deletes the index copy in the repository and redefines it so that keys reference the additional CIAI columns.

For example:

- The S_CONTACT table has Base Column A with Index A that has two columns as keys, LAST_NAME and FST_NAME.
- You then select the column LAST_NAME for CIAI queries and specify the Copy All index strategy.
- The Case Insensitivity Wizard defines the column LAST_NAME_CI and a CIAI index for the new column.
- The wizard also defines Index B for Base Column A by copying Index A and specifying LAST_NAME_CI and FST_NAME as keys.
- You then select the column FST_NAME for CIAI queries.
- As part of configuring FST_NAME for CIAI queries, the wizard does the following in the repository:
  - Defines a new column FST_NAME_CI.
  - Deletes Index B on Base Column A and redefines it with the keys LAST_NAME_CI and FST_NAME_CI.

Operation: Inactivating CIAI Configuration

The Operation field in the configuration syntax controls whether the columns and indexes created by the Case Insensitivity Wizard are active. The available settings are On and Off. The default is On.

Use the Operation field to deactivate the CIAI configuration of columns. When you run the Case Insensitivity Wizard against a column and specify Operation indicates Off, the wizard does the following:

- Inactivates the CIAI column created on the base column.
- Inactivates the CIAI index created for the CIAI column.
- For indexes that the base column participates in, sets the related CIAI indexes to inactive.

Note that the Case Insensitivity Wizard does not delete CIAI columns or CIAI indexes in the repository.

You cannot use Operation indicates Off to set the Default Insensitivity property on a column from DB Case & Accent to None.

Case Insensitivity Wizard Defaults

The only required inputs for the Case Insensitivity Wizard are Table Name and Column Name. If you omit the other fields, then the wizard uses the default settings.

Methods

The Case Insensitivity Wizard uses the following default settings for the methods:

- In the repository, if the Force Case property is set on the column, then the wizard uses the Force Case method.
- If the Force Case property is not set, then the wizard uses the Database method.

Index Strategy

The Case Insensitivity Wizard uses the following defaults for index strategy:

- If the method is Force Case, then the wizard sets the index strategy to None.
- If the method is Database, and the base column does not have indexes defined on it, then the wizard sets the index strategy to None.
- If the method is Database, and the base column has indexes defined on it, then the wizard uses the Copy All index strategy.

If the Case Insensitivity Wizard uses None as an index strategy, then the wizard does not define new columns or indexes. It sets Default Insensitivity to DB Case & Accent.

This setting means that the Case Insensitivity Wizard has the following default behaviors:

- When the Force Case property is set on a column, the wizard does not define columns or indexes.
- If the column has no indexes defined on it, then the wizard does not define columns or indexes.

Note that in both cases, the Case Insensitivity Wizard accepts the column as eligible but does not define columns or indexes. These default behaviors define the implicit eligibility requirements.

Operation

If you omit Operation, then the Case Insensitivity Wizard sets Operation to On, regardless of the method or index strategy.

Running the Case Insensitivity Wizard Multiple Times

You can run the Case Insensitivity Wizard multiple times. Typically, you would do this to tune the CIAI configuration of columns. If you have already run the Case Insensitivity Wizard once to configure columns, then you might run it again to perform additional tasks. The following table lists operations that you can perform when running the Case Insensitivity Wizard again:

Task	How to Run the Case Insensitivity Wizard
Configure new columns to support CIAI queries.	You can use either an input file, or you can select files manually.
Inactivate CIAI configuration for specified columns.	Run the Case Insensitivity Wizard using an input file that specifies the desired columns. For each column, specify Operation indicates Off.
Run the Case Insensitivity Wizard to change the Default Insensitivity property from None to DB Case & Accent for eligible columns without indexes.	Start the Case Insensitivity Wizard from the Case Insensitivity menu (navigate to Tools, Utilities, then Case Insensitivity) and select the mode Enable for all unindexed columns.
For columns already configured, change the method from Force Case to Database.	You can use either an input file, or you can select files manually.
For columns already configured, change the method from Database to Force Case.	Run the Case Insensitivity Wizard using an input file that specifies the desired columns. For each column, specify Operation indicates Off. This deactivates the CIAI column and CIAI indexes. Verify that the base columns have the Force Case property set.
For columns already configured, change the index strategy from Single to Copy All.	Run the Case Insensitivity Wizard with an input file that specifies this change.
For columns already configured, change the index strategy from Copy All to Single.	Run the Case Insensitivity Wizard using an input file that specifies the desired columns. For each column, specify Operation indicates Off. This deactivates the CIAI column and CIAI indexes. Then run the Case Insensitivity Wizard on the same base columns, with the Database method and the Single index strategy. This activates the index on the CIAI columns.

Column and Index Naming Convention

The Case Insensitivity Wizard uses the same naming conventions as the Siebel Enterprise Integration Manager Wizard when creating CIAI column names and index names. The naming conventions used by the wizard are fixed and cannot be overridden.

Column Names

The Case Insensitivity Wizard defines CIAI column names by appending _CI to the parent column name. For example, if the parent column is LAST_NAME, then the CIAI column would be LAST_NAME_CI.

Index Names

The Case Insensitivity Wizard defines CIAI index names by adding a string to the base table name:

```
base_table_name_C#
```

In this table name, # is an integer incremented starting at 1 as needed to create a unique name.

An example of a CIAI index created on table S_CON_ADDR is S_CON_ADDR_C1.

Truncating Names

The default length for column names and index names is 30 characters. You can limit the length of names to 18 characters (to conform to the requirements of IBM DB2), prior to running the CIAI, by selecting the "Limit schema object names to 18 characters" option by navigating to Tools, View, Options, and then Database.

If CIAI column names or index names exceed the maximum length, then the Case Insensitivity Wizard truncates the column base name or the table base name (for indexes) using the following strategy:

- Deletes underscores one at a time, beginning with the first underscore.
- Deletes vowels one at a time, beginning with the last vowel.
- Deletes characters (letters, numbers, and so on) one at a time, beginning with the last character.

The Case Insensitivity Wizard does not truncate prefixes or postfixes.

Column and Index Name Uniqueness

After truncating a column or index name, the name might not be unique. When this occurs, the Case Insensitivity Wizard modifies the truncated column or index name by truncating the last character in the base column name or base table name. The wizard replaces the truncated character with an integer, starting at 1.

If this modification does not create a unique name, then the wizard increments the integer. If the integer becomes two digits or larger, then the wizard truncates the name to make room for the digits. This process maintains the overall string length.

Error Reporting

The Case Insensitivity Wizard reports errors in a pane in the wizard. The error listing provides enough information so that you can identify the column and the cause of the problem.

You then have two options:

- You can correct the errors and rerun the wizard.
- You can ignore the errors.

When the wizard configures columns, it skips all columns that have errors.

When the Case Insensitivity Wizard reports errors, you can export them to a text file for reference. Errors usually fall into one of the following categories:

- **Input file syntax errors.** These include punctuation and use of improper configuration options.
- **Table and column eligibility problems.** These errors occur when you choose tables and columns of types that the wizard does not support.
- **Project not locked.** You must lock the tables you want to configure before running the wizard. The wizard displays the list of projects that must be locked.